IBM InfoSphere Data Replication
Version 10.1.3

SQL Replication Guide and Reference

SC19-3638-00

IBM InfoSphere Data Replication
Version 10.1.3

SQL Replication Guide and Reference

SC19-3638-00

Note

Before using this information and the product that it supports, read the information in "Notices and trademarks" on page 469.

Contents

Chapter 1. Planning for SQL Replication

When planning for SQL Replication, you might need to consider planning for migration, memory, storage, conflicts, source systems, code page conversion, and performance.

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.

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.

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.

z/OS Linux UNIX Windows

When replication reads log records it uses a memory buffer. 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. The default size of the buffer on Linux, UNIX and Windows operating systems is fifty 4 KB pages.

System i

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.

z/OS Linux UNIX Windows

- 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.

System i

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

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.

System i You also need to consider the log space (journal receivers space) of the target tables. Because journal receivers for target tables on System i® can be created with the **MNGRCV(*SYSTEM)** and **DLTRCV(*YES)** parameters, and because you need to journal only the after-image columns, use the following formula to estimate the volume of the journal receivers for the target tables:

```
journal_receiver_volume=target_table_row_length X journal_receiver_threshold
```

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 prune the tables by 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 are captured for the source table within that contingency time period.

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

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

Example

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

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

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

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.

z/OS

Spill files go to virtual I/O (VIO).

Linux UNIX Windows

Spill files are always on disk. One file per transaction is created in the **capture_path** directory.

System i

Spill files are created in library QTEMP, one spill file for each registration that needs a spill file.

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).

z/OS Linux UNIX Windows 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.

System i 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 (z/OS, Linux, UNIX, Windows)

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 codes[1] less than or equal to 49999 as the key range), but can read all sales records.

Fragmentation by time
: Design your application so that the table can be updated only during

1. United States postal codes.

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.

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 existing triggers with Capture triggers

The Capture trigger logic is in the SQL script generated by the Replication Center when you register a source.

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

If a trigger already exists on the table that you want to register for replication and that trigger has the same name as the one that is in the generated script, you'll receive a warning when the script is generated. Do not run the generated script because the RDBMS might overwrite the existing trigger. Determine how you want to merge the 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 synch point. The locks on the CCD tables are held only until the Apply program sets its synch point, not through the entire Apply cycle. Applications that need to update the source table must wait until the Apply program unlocks the CCD table.

Required changes for Sybase triggers on little-endian platforms

When setting up Capture triggers from Sybase servers, you might need to change the triggers if the operating system on which Sybase runs uses little-endian byte order.

The Capture triggers rely on the @@dbts function in Sybase to update synchpoint values. The function returns the value of the current timestamp for the database. This value is not returned correctly if Sybase server is running on an operating system that uses little-endian byte order, for example Windows NT32. In Sybase on little-endian platforms, timestamp column values are displayed as big-endian values, while the result of the @@dbts function is still displayed as a native little-endian value.

Because of this issue, you must manually update the Capture trigger script that is generated by the replication administration tools if your Sybase server is on this type of operating system. In the script, change @@dbts to the following:

```
reverse(substring(@@dbts,1,2)) + 0x0000 + reverse(substring(@@dbts,5,4))
```

Code page conversion planning

Replication components are database applications that rely on the DB2 databases on various operating systems to handle conversion of data that uses different code pages.

Replication components work with data by using SQL SELECT, INSERT, UPDATE, and DELETE statements.

Replication 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 conversion. 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 conversion.

If you plan to replicate between databases with differing code pages, check the *IBM Information Management Software for z/OS Solutions Information Center* or *DB2 Information Center* to determine if the code pages you have are compatible. For example, if you are using DB2 for Linux, UNIX, and 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 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.

Code pages for SQL Replication

The code page 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 might 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 System i 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 a System i table with multiple columns using different code pages, those columns cannot be replicated to a single DB2 for Linux, UNIX, and Windows database unless all the code pages are compatible.

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

Replication planning for DB2 for z/OS

SQL replication for DB2 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 for z/OS Version 8 or later in new-function mode.
- Run the Capture, Apply, and Monitor servers under DB2 for z/OS Version 8 or later in new-function mode

Restriction: If you want to replicate to or from DB2 for iSeries®, you must use schema names that are 30 bytes or shorter. Replication among DB2 for z/OS and DB2 for Linux, UNIX, and Windows platforms supports 128-byte schema names.

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. Authorization requirements for SQL Replication

To use the SQL Replication programs, you need to ensure that user IDs that operate the replication programs or use the replication tools have the correct authorization on local and remote systems.

Authentication requirements on Linux, UNIX, and Windows

SQL Replication does not require you to use any specific type of authentication. You should use the compatible authentication type that best meets your business needs.

The following list provides more detail:

- SQL Replication is a database application program.
- The underlying DB2 client-server facilities are as transparent to SQL Replication as they are to any database application.
- SQL Replication connects to DB2 databases using the traditional ID and password combination. You can use any authentication type that is compatible with this.
- SQL Replication has no requirement on authentication type other than this.

Authorization requirements for administration

To set up replication, you run generated SQL to create objects, and bind plans and create SQL packages (System i). 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.

z/OS Linux UNIX Windows

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 by 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.

System i

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 System i 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_10_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.

Windows

The user ID that runs the Capture program must be authorized to create global objects.

System i

Use the Grant DPR Authority (**`GRTDPRAUT`**) command to authorize a user to run the Capture program on a local system. If you are replicating between only System i 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 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_10_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.

Windows

The user ID that runs the Apply program must be authorized to create global objects.

System i

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 System i 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.

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.

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.

For Oracle sources, you need to grant SELECT privilege on the *remote_schema*.SEQUENCE002 sequence object, where *remote_schema* is the remote schema under which the control tables are created on Oracle. The sequence object is created as part of creating the Capture control tables for an Oracle source and is used along with Capture triggers to populate the CCD table.

Managing user IDs and passwords for remote servers (Linux, UNIX, Windows)

Replication and Event Publishing require a password file in some cases to store user IDs and passwords for connecting to remote servers.

About this task

A password file is required in the following cases:

- The Apply program requires a password file to access data on remote servers (the Capture program does not require a password file).
- The Q Apply program requires a password file to connect to the Q Capture server for Q subscriptions that use the EXPORT utility to load targets.
- The Q Capture program requires a password file to connect to multiple-partition databases.
- If the Q Capture program runs remotely from the source database or the Q Apply program runs remotely from the target database, the programs require password files to connect to the remote database.
- The **asntdiff** and **asntrep** commands require password files to connect to databases where the utilities are comparing or repairing table differences.
- The Replication Alert Monitor requires a password file to connect to any Q Capture, Capture, Q Apply, or Apply server that you want to monitor.

Important note about compatibility of password files: Password files that are created by the **asnpwd** command starting with Version 9.5 Fix Pack 2 use a different encryption method and cannot be read by older versions of the replication programs and utilities. If you share a password file among programs and utilities that are at a mixed level, with some older than these fix packs, do not recreate the password file by using an **asnpwd** command that is at these fix packs or newer. Replication programs and utilities at these fix packs or newer can continue to work with older password files. Also, you cannot change an older password file to use the later encryption method; you must create a new password file.

In general, replication and Event Publishing support the following scenarios:

- Creating a password file with one version and using it with a newer version. For example, you can create a password file under V8.2 and use it with V9.1 and V9.5.
- Creating a password file with one fix pack and using it with a newer fix pack within the same version. For example, you can create a password file with V9.1 Fix Pack 3 and use it with V9.1 Fix Pack 5.
- Creating a password file on one system and using it on another system as long as the following criteria are met:
  - The systems use the same code page.
  - The systems are all 32 bit or all 64 bit.

Encrypted password files are not supported for x64 Windows until 9.5 Fix Pack 2 or later.

Procedure

To manage user IDs and passwords for remote servers, follow these guidelines:

- Create an encrypted password file for replication and event publishing programs that are running on Linux, UNIX, and Windows by using the **asnpwd** command. The password file must be stored in the path that is set by the following parameters:

Table 1. Password file requirements

Program	Parameter
Apply	**apply_path**
Q Apply	**apply_path**
Q Capture	**capture_path**
Replication Alert Monitor	**monitor_path**
asntdiff or asntrep command	**DIFF_PATH**

- If the Q Apply program and Replication Alert Monitor are running on the same system, they can share the same password file. If you want the programs to share a password file, specify the same path and file name for the programs, or use symbolic links to share the same password file in the different directories.
- The Replication Center does not use the password file that is created with the **asnpwd** command to connect to remote servers. The first time that the Replication Center needs to access a database or subsystem, you are prompted for a user ID and password, which is stored for future use. You can use the Manage Passwords and Connectivity window to store user IDs for servers or systems, as well as to change the IDs that you stored and to test connections. To open the window, right-click the **Replication Center** folder and select **Manage Passwords for Replication Center**.

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.

z/OS For more detail about configuring servers on z/OS, see Replication installation and customization for z/OS.

Required: Setting DATA CAPTURE CHANGES on DB2 source tables and DB2 for z/OS system tables

You must set the DATA CAPTURE CHANGES attribute on any table that you want to replicate. Also, on DB2 for z/OS Version 9 and later, you must set DATA CAPTURE CHANGES on the SYSIBM.SYSTABLES, SYSIBM.SYSCOLUMNS, and SYSIBM.SYSTABLEPART system catalog tables.

About this task

Setting DATA CAPTURE CHANGES on source tables prompts DB2 to log SQL changes in an expanded format that is required for replication. The replication administration tools will generate the DDL to alter the table if this option is not set. However, you can set it when creating tables or alter the table yourself.

For DB2 for z/OS system tables, setting DATA CAPTURE CHANGES enables detection and replication of changes to the structure of source tables such as addition of new columns or changes in column data types.

Note: If the replication source is DB2 for z/OS Version 9 or later, the Capture program stops if DATA CAPTURE CHANGES is not set on the SYSIBM.SYSTABLES, SYSIBM.SYSCOLUMNS, and SYSIBM.SYSTABLEPART system catalog tables.

Turning on DATA CAPTURE CHANGES for a table introduces a small amount of extra logging. When this option is set for a table, DB2 logs both the full before and after images of the row for each update. When the option is not set, DB2 logs only the columns that changed because this amount of logging is all that is needed for DB2 recovery. The amount of additional logging is proportional to row size, but only for those tables for which DATA CAPTURE CHANGES is on.

Procedure

Use the CREATE TABLE or ALTER TABLE statement to set DATA CAPTURE CHANGES on replication source tables. For DB2 for z/OS system catalog tables, this step is performed for you when you create control tables using the Version 10 Replication Center or ASNCLP program. You can also use the following statements:

```
ALTER TABLE SYSIBM.SYSTABLES DATA CAPTURE CHANGES;
ALTER TABLE SYSIBM.SYSCOLUMNS DATA CAPTURE CHANGES;
ALTER TABLE SYSIBM.SYSTABLEPART DATA CAPTURE CHANGES;
```

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 System i servers from Windows

You can administer replication on System i 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 a System i server from a DB2 for Windows workstation:

1. Log on to the System i server and locate the relational database:
   a. Log on to the System i 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 System i database in DB2 for Windows:
   a. From a Windows command prompt, enter **`db2cmd`**. 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 System i system, and *rdb_name* is the name of the System i relational database that you found in Step 1.

3. In the command window, issue the following command:

   ```
   db2 terminate
   ```

4. Ensure that the System i user profile that you will use to log on to your System i system uses CCSID37:
   a. Log on to the System i 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 System i system type:

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

5. Make sure that DB2 for Windows and DB2 for System i 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 by using a server mapping.

   Restriction: The Replication Center and ASNCLP command-line program do not support creating control tables or target tables in Oracle databases if the server mapping has two-phase commit enabled.
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.

Creating control tables for SQL Replication

You can use the ASNCLP command-line program or Replication Center 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:
	`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;`

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: • **Create Capture Control Tables** • **Create Capture Control Tables** > **Quick** • **Create Apply Control Tables** • **Create Apply Control Tables** > **Quick**

Creating control tables (System i)

Replication control tables are created automatically when you install DB2 DataPropagator for System i. 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 System i, use the Create DPR Tables (**CRTDPRTBL**) command.

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

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 it you want to work with UNICODE and EBCDIC encoding schemes separately or if you want to run more than one instance of the Capture program on a subsystem.

System i 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.

Creating control tables in a multiple-partitioned database

When you create Capture control tables in a multiple partitioned database, the control tables should be placed in a single-partition table space that is on the catalog partition.

Typically, you first create a single-partition table space, and then specify that table space when you use the Replication Center or ASNCLP command-line program to create the control tables.

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.

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 UNIX	`export DB2INSTANCE=`*`db2_instance_name`*
Windows	`SET DB2INSTANCE=`*`db2_instance_name`*

2. If you created the source database with a code page other than the default code page value, set the DB2CODEPAGE environment variable to that code page. **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 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. Check the "Log retain for recovery status" value in the database configuration. If it is set to NO, turn on archival logging by changing the LOGARCHMETH1 database configuration parameter value to a value other than OFF.

For multiple database partition environments, every partition must be set up to allow roll-forward recovery for every partition that the Capture will capture changes from.

2. 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 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.
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 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 (System i)

You need to create packages using the **CRTSQLPKG** command in some cases on System i.

About this task

Use this command to create packages in the following cases:
- When you use 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 you use 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 System i systems or between a System i system and some other operating system (such as Linux, UNIX, or Windows).

For information about using the **CRTSQLPKG** command, see *DB2 for i5/OS SQL Programming*.

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

Creating SQL packages for the Apply program (System i)

You must create SQL packages for the Apply program on System i 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:

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

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

Creating SQL packages for the Replication Analyzer (System i)

You must create SQL packages for the Replication Analyzer on System i 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:

```
CRTSQLPKG 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 (System i)

After you create SQL packages on System i, 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 System i 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(ASN/package_name) OBJTYPE(*SQLPKG) USER(subscriber_name) AUT(*OBJOPR *EXECUTE)`

Method	Description
GRANT SQL statement	Use SQL to connect to the source database and run the GRANT SQL statement: `CONNECT TO data_server_RDB_name` `GRANT EXECUTE ON PACKAGE ASN/package_name TO subscriber_name`
`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 Replication installation and customization 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 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.

Replication of partitioned tables: Version 9.7 Fix Pack 1 or earlier (Linux, UNIX, Windows)

SQL Replication supports DB2 tables that are partitioned by range (using the PARTITION BY clause of the CREATE TABLE statement). These tables are sometimes known as range-partitioned tables.

Version and fix pack requirements exist for the Capture program if a source table is partitioned by range. This topic covers these requirements and support for replication of partitioned tables when your replication programs are at Version 9.7 Fix Pack 1 or earlier. If only target tables (no source tables) are partitioned by range, then SQL Replication has no version or fix pack requirements specific to these tables.

To capture changed data for range-partitioned tables, your Capture program must be at Version 9.7 or later. It can capture changes from range-partitioned tables on

DB2 V9.1, V9.5, or V9.7. However, restrictions exist for range-partitioned tables earlier than V9.7. The restrictions are also discussed in this topic.

Partitioned tables use a data organization scheme in which table data is divided across multiple storage objects, called data partitions or ranges, according to values in one or more table partitioning key columns of the table.

SQL Replication treats all data partitions of a source table as a single table. For example, when you register a partitioned table, you specify the entire table rather than one or more data partitions of the table. All row operations for the table, regardless of the data partition at which they occur, are replicated.

You can perform several alterations on a partitioned table, including adding a data partition, attaching a data partition, or detaching a data partition. These ALTER operations on the source table are not replicated to the target. You must alter the target table independently of the source table if you want to maintain an identical partitioning scheme.

SQL Replication treats these ALTER operations differently:

ADD
Adds a new, empty data partition to the source table. If you require the new data partition at the target, you must manually add it. Capture program behavior and the procedure that you need to follow depend on the release of your DB2:

Version 9.7 or higher
Add the data partition at the target before adding it at the source. Capture automatically begins replicating changes to the data partition.

Version 9.5 or 9.1
Capture does not recognize the addition of the data partition until the program is reinitialized or stopped and restarted. Add the data partition at both the source and target before restarting Capture. Do not change data in the source data partition until Capture is restarted.

ATTACH
Creates a new data partition at the source by using an existing table. The ATTACH operation is not replicated, and the data in the new data partition is not replicated to the target. If you require the new data partition at the target you must manually add it. If you require the attached data at the target, you must manually load it into the target.

Note: If the Capture program is stopped when a data partition is attached, rows that are inserted, updated, or deleted on the table before it is attached as a data partition are replicated. If Capture is running when the data partition is attached, these rows are not replicated.
To ensure consistent behavior, before you attach a table as a new data partition, set the DATA CAPTURE CHANGES clause for the table to OFF if you need to make any changes to the table. For example, the following statements create a table, insert values into the table, and then attach the table as a data partition to an existing partitioned table:

```
db2 create table temp1 like t1;
--  NOTE: data capture changes is off by default
db2 insert into temp1 values (44,44);
-- NOTE: Turn on data capture changes after insert/update/deletes
-- and before attach partition
db2 alter table temp1 data capture changes;
```

```
db2 alter table t1 attach partition part4 starting from 41
ending at 50 from temp1;
db2 set integrity for t1 allow write access immediate checked;
```

DETACH
Turns an existing data partition into a separate table. The DETACH operation is not replicated. The data that is deleted from the source table by the DETACH operation is not deleted from the target table. If you need to change the target data partition into a separate table, you need to do so manually.

Note: DB2 logs updates that cause rows to move across data partitions as delete-insert pairs. The Capture program also treats these updates as deletes and inserts in the CD table (similar to the behavior of CHG_UPD_TO_DEL_INS=Y in the IBMSNAP_REGISTER table).

Replication of partitioned DB2 tables: Version 9.7 Fix Pack 2 or later (Linux, UNIX, Windows)

SQL Replication supports DB2 tables that are partitioned by range (using the PARTITION BY clause of the CREATE TABLE statement). These tables are sometimes known as range-partitioned tables.

Version and fix pack requirements exist for the Capture program if a source table is partitioned by range. This topic covers these requirements and support for replication of partitioned tables when your replication programs are at Version 9.7 Fix Pack 2 or later. If only target tables (no source tables) are partitioned by range, then SQL Replication has no version or fix pack requirements specific to these tables.

To capture changed data for range-partitioned tables, your Capture program must be at Version 9.7 or later. It can capture changes from range-partitioned tables on DB2 V9.1, V9.5, or V9.7. However, restrictions exist for range-partitioned tables earlier than V9.7. The restrictions are also discussed in this topic.

Important: If your replication programs are at Version 9.7 Fix Pack 2 or later and you plan to replicate range-partitioned tables, you must run the Version 9.7 Fix Pack 2 migration script, `asncapluwv97fp2.sql`. The script adds a new control table, IBMQREP_PART_HIST, to help the replication programs handle data partition changes such as add, attach, or detach. The script is located in the `samples/repl/mig97/sql/` directory. The Capture program does not use the IBMQREP_PART_HIST table for partitioned source tables on DB2 Version 9.5 or Version 9.1.

Partitioned tables use a data organization scheme in which table data is divided across multiple storage objects, called data partitions or ranges, according to values in one or more table partitioning key columns of the table.

SQL Replication treats all data partitions of a source table as a single table. For example, when you register a partitioned table, you specify the entire table rather than one or more data partitions of the table. All row operations for the table, regardless of the data partition at which they occur, are replicated.

You can perform several alterations on a partitioned table, including adding a data partition, attaching a data partition, or detaching a data partition. These ALTER operations on the source table are not replicated to the target. You must alter the target table independently of the source table if you want to maintain an identical partitioning scheme.

SQL Replication treats these ALTER operations differently:

ADD Adds a new, empty data partition to the source table. If you require the new data partition at the target, you must manually add it. Capture program behavior and the procedure that you need to follow depend on the release of your DB2:

Version 9.7 or higher
Add the data partition at the target before adding it at the source. Capture automatically begins replicating changes to the data partition.

Version 9.5 or 9.1
Capture does not recognize the addition of the data partition until the program is reinitialized or stopped and restarted. Add the data partition at both the source and target before restarting Capture. Do not change data in the source data partition until Capture is restarted.

ATTACH
Creates a new data partition at the source by using an existing table. The ATTACH operation is not replicated, and the data in the new data partition is not replicated to the target. If you require the new data partition at the target you must manually add it. If you require the attached data at the target, you must manually load it into the target.

DETACH
Turns an existing data partition into a separate table. The DETACH operation is not replicated. The data that is deleted from the source table by the DETACH operation is not deleted from the target table. If you need to change the target data partition into a separate table, you need to do so manually.

Note: DB2 logs updates that cause rows to move across data partitions as delete-insert pairs. The Capture program also treats these updates as deletes and inserts in the CD table (similar to the behavior of CHG_UPD_TO_DEL_INS=Y in the IBMSNAP_REGISTER table).

Running DB2 Query Patroller in a SQL Replication environment

If you set up replication in an environment that also runs DB2 Query Patroller, you need to take special steps to ensure that replication activities are not compromised.

About this task

Query Patroller monitors the cost of dynamic queries that are issued against a database. If you run the Query Patroller client and the cost of a query exceeds a threshold set by the database administrator, the query is trapped and a message issued.

The replication components can issue many dynamic queries. If Query Patroller is enabled and the client is installed where the replication components are running, these situations might occur:

- Dialog messages are issued on the client system when a dynamic query from replication exceeds the defined threshold.
- If Query Patroller detects an error, the replication components can receive a nonzero SQLCODE from Query Patroller. Most of these SQLCODEs are in the range between SQL29000N and SQL29999N.

Procedure

To avoid these situations, take one of the following actions:
- Run the replication components from a DB2 client that does not have the Query Patroller client enabled.
- Disable Query Patroller. For example, you can disable it for a given database by setting the DYN_QUERY_MGMT parameter to 0 (DISABLE). DYN_QUERY_MGMT is a database configuration parameter that determines whether Query Patroller is enabled for a given database.

Setting up journals (System i)

DB2 DataPropagator for System i 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 System i 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 System i, 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 (System i)

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 System i 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 by 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 *System i Programming: Performance Tools Guide* for more information.
3. Start journaling the source table by using the Start Journal Physical File (**STRJRNPF**) command, as in the following example:

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

   Specify the name of the journal that you created in Step 2. The Capture program requires a value of *BOTH for the IMAGES parameter.
4. Change the source table journaling setup:
   a. Use IMAGES(*BOTH) to make sure that the source table is journaled for both before- and after-images.
   b. Make sure that the journal has the following attributes: MNGRCV(*SYSTEM) and DLTRCV(*YES).
   c. Make sure that the journal has the MINENTDTA(*NONE) attribute.
   d. For journals on remote systems, specify the MNGRCV(*SYSTEM), DLTRCV(*YES), and MINENTDTA(*NONE) attributes on the source journal. To define the remote journal, specify the DLTRCV(*YES) attribute on the **ADDRMTJRN** command.

Managing journals and journal receivers (System i)

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 (System i):

You should let the System i 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.

Remote journaling with different system times (System i):

If the system time (QTIME) of the source and target systems do not match in a replication environment that uses remote journaling, you need to take precautions in the initial handshake between the Capture and Apply programs.

When replicating with remote journals, the Capture and Apply programs operate as if the source system is local when it is not. If possible, make sure that the QTIME of the source and target systems match.

If you cannot match system times, take the following precautions:

- If the source system time is ahead of the target system time, make sure that PTF SE23500/SI21622 is installed.
- If the source system time is behind the target system time, the Capture program starts receiving changes based on the target system time. Wait for the source system time to become greater than the time of the full refresh. Then make dummy changes and check to see if the changes are replicated before you allow applications to update the source table.

In general, when the source and target system times are different, avoid operations that will cause a full refresh again, such as CLRPFM and CPYF with *REPLACE.

After the Capture program has started processing changes for the source table, you could set the column DISABLE_REFRESH in the IBMSNAP_REGISTER table to 1 for that table. If a full refresh is needed, the Apply program fails and you could coordinate the full refresh when you are ready to perform the handshake in a controlled manner by setting DISABLE_REFRESH to 0.

Avoiding unwanted full refreshes and other problems caused by a time mismatch (System i):

When the system times at the source and target servers do not match, unwanted full refreshes can occur and journal entries might be missed. You can take steps to avoid these problems.

Both problems have the same root cause. When the Apply program signals that a full refresh has begun, a trigger on the system where Capture is running fills in the FR_START_TIME value in the IBMSNAP_REG_EXT table (in the remote journal case the trigger is on the target system). The time that the trigger uses is the current timestamp of the machine where the trigger is running. The trigger does not have the ability to get the timestamp from the source system.

The potential problems that are caused by time mismatches fall into two categories and are described in the following sections:
- "Source system clock ahead of target system clock"
- "Source system clock behind target system clock"

Source system clock ahead of target system clock

When the source system clock is ahead of the target system clock, a Clear Physical File Member (CLRPFM) command for the source table causes multiple full refreshes.

When a CLRPFM (or similar journal entry) comes in, a full refresh is started. In remote journaling, the timestamp of the target system is put into the FR_START_TIME field. If the source system clock is ahead of the target system clock, the FR_START_TIME value will be less than the time of the CLRPFM on the source system.

Capture uses the FR_START_TIME as the time when journal entries should be picked up. Because the target is behind the source, Capture will see the CLRPFM again, and cause another full refresh to happen. This loop will continue until the time of the full refresh on the target system is beyond the journal entry time of the CLRPFM.

To avoid this problem, follow these steps:
1. Stop the Apply program but keep the Capture program running.
2. Issue the CLRPFM command at the source.
3. When the timestamp on the target system is later than the time that the CLRPFM command was issued on the source system, start Apply.

Source system clock behind target system clock

When the target system clock is ahead of the source system clock, Capture uses the FR_START_TIME to look for journal entries that are later than expected. Consider this scenario where the target system is one hour ahead of the source system:
- The full refresh happens at 05:00 on the target system, which is 04:00 on the source system.
- Because the FR_START_TIME is set to 05:00, Capture does not see any journal entries from the source until the source system clock becomes 05:00.
- Capture does not see the journal entries between 04:00 and 05:00 on the source system.

To avoid this problem, follow these steps:
1. Make sure that the Capture and Apply programs are stopped.

2. Start Capture. This step makes Capture aware of the new tables that are waiting for a full refresh to occur.
3. Wait until the journal jobs start.
4. Stop Capture and wait for the program to completely end.
5. Start Apply and let the full refresh occur.
6. Take one of these steps:
   - Make sure that no data goes into the source table between the time when the full refresh is done and the time that the source system clock catches up to the target system clock.
   - Update the FR_START_TIME value to match the clock time on the source system when the full refresh started.
7. Start Capture.

Changing definitions of work management objects (System i):

System i

You can alter the default definitions for the three types of work management objects that are created during installation of DB2 DataPropagator for System i, 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 2 lists the objects that are created.

Table 2. Work management objects

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

If you create your own subsystem description, you must name the subsystem QZSNDPR and create it in a library other than QDP4. See *System i Work Management Guide* (SC41-5306) for more information about changing these definitions.

Specifying user management of journal receivers (System i):

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 *System i Backup and Recovery*.

Delete journal receiver exit routine (System i):

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 System i, 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 DLTOPT(*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 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)

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 System i, 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

System i

- 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.

Linux UNIX Windows

- Replication is supported from multiple-partition databases. There is no limit to the number of partitions that replication supports.

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

System i

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

Linux UNIX Windows

- DB2 tables that your application maintains
- Catalog tables (for full-refresh-only replication)
- Materialized query tables
- External CCD tables
- Tables that are partitioned with the DISTRIBUTE BY clause of the CREATE TABLE statement
- Tables that are partitioned by range (using the PARTITION BY clause of the CREATE TABLE statement)
- Compressed tables

You can register the same table multiple times by 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.
	`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;`

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.

Method	Description
System i ADDDPRREG system command	Use the ADDDPRREG command to register a table on System i. 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: `ADDDPRREG SRCTBL(HR/EMPLOYEE) CAPCTLLIB(BSN) CDLIB(HRCDLIB) CDNAME(CDEMPLOYEE)`

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.

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	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:
	• 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.tesnk
	• All columns in repldba.tesnk are registered with after images
	`SET SERVER CAPTURE TO DB FEDORADB NONIBM SERVER V9ORA`
	`ID repldba PASSWORD "passw0rd";`
	`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;`
	
	The CONDENSED OFF option is required for federated sources.

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

When 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).

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.

z/OS Linux UNIX Windows

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

System i

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.

For columns that have before-images defined, replication limits column names to 127 characters because the entire column name can have only 128 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 (the default is X) to target columns and each column name must be unique, you cannot use column names that are longer than 127 characters. For tables that you do not plan to replicate, you can use longer column names, but consider using 127-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 by 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 System i 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-System i) 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.

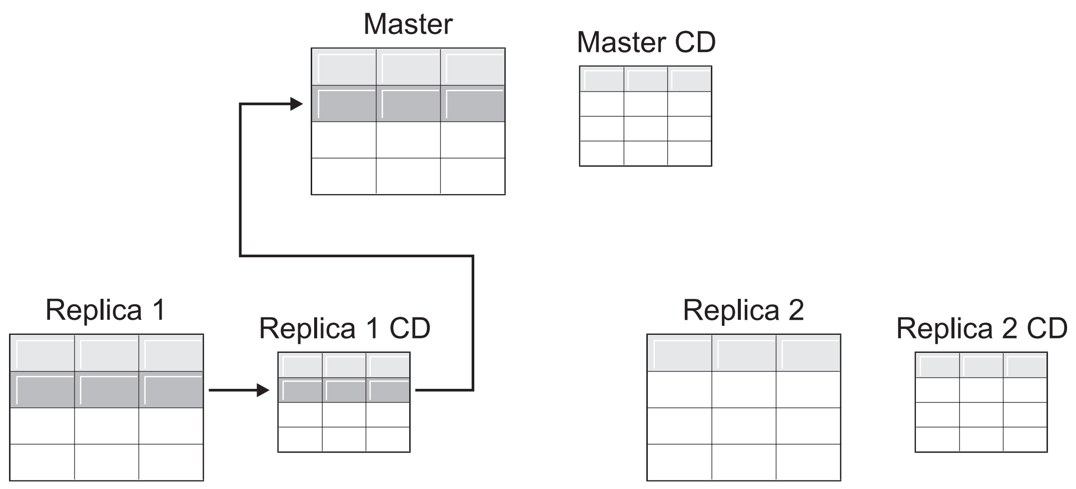


Figure 1. Recapture option for replicas that are mutually exclusive partitions of the master. When you have multiple replicas that do not subscribe to the same data in the master, you do not need to use the recapture option for any of the tables.

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.

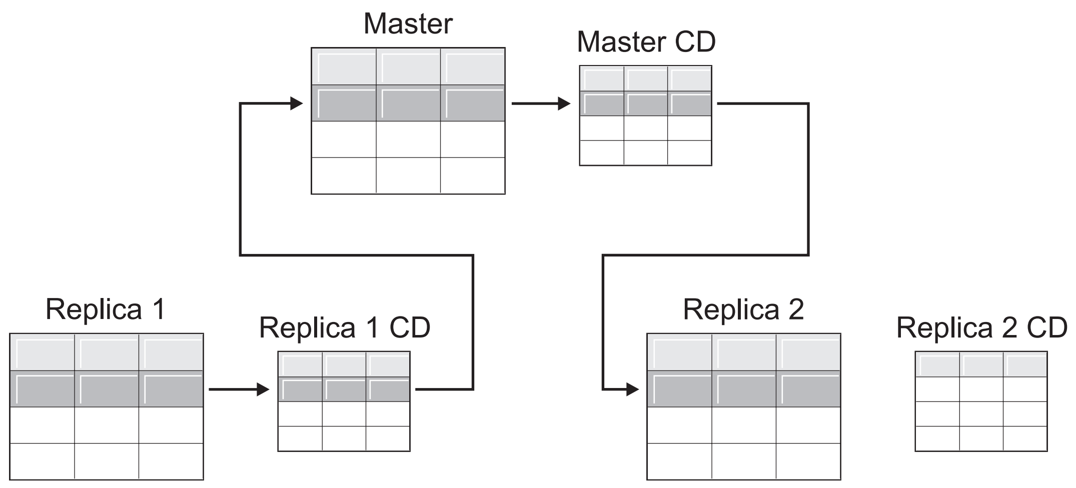


Figure 2. Recapture option for masters that replicate changes to multiple replicas. When you have multiple replicas that subscribe to the same data in the master, you can use the recapture option at the master so that changes that occur at one replica are recaptured at the master and forwarded to the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

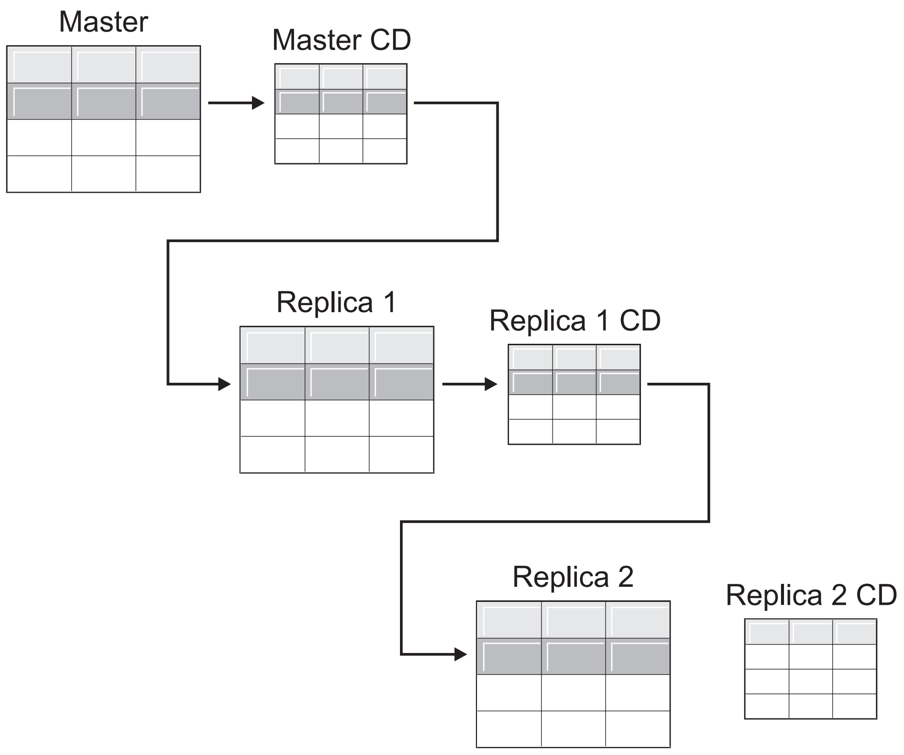


Figure 3. Recapture option at tier 2 allows changes at tier 1 to be replicated down to tier 3. When you have a replica table that acts as a middle tier in a multi-tier configuration, you can use the recapture option at the replica so that changes that occur at the master are recaptured at the replica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

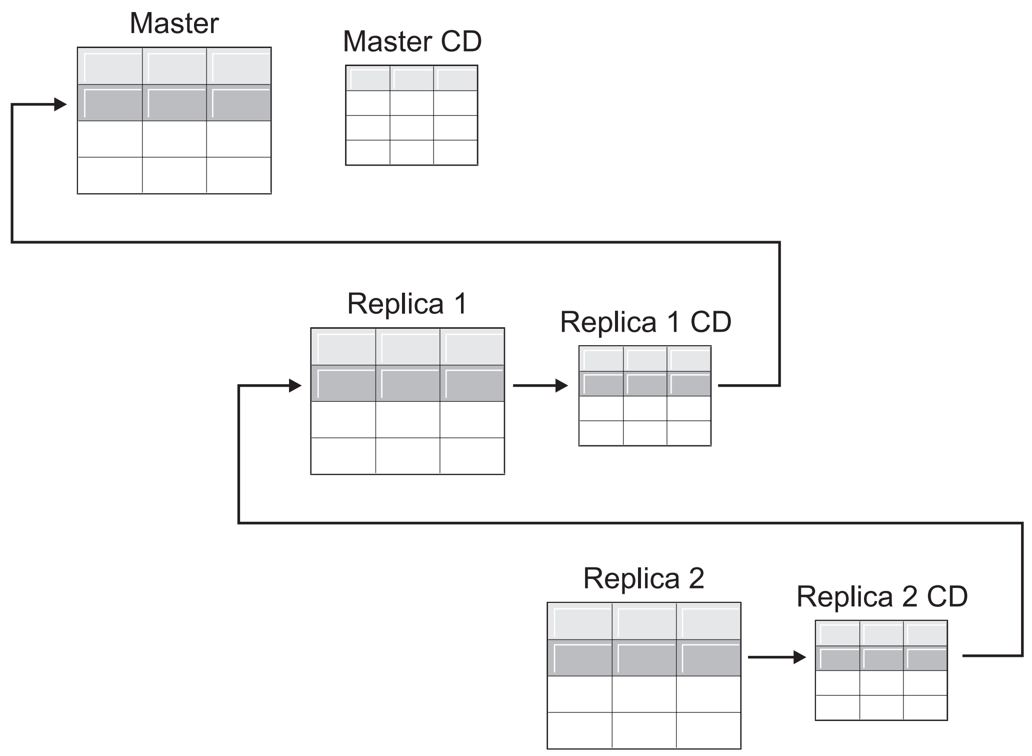


Figure 4. Recapture option at tier 2 allows changes at tier 3 to be replicated up to tier 1. When you have a replica table that acts as a middle tier in a multi-tier configuration, you can use the recapture option at the replica so that changes that occur at the replica in the subsequent tier are recaptured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

When registering System i 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 a System i database server that is separate from the System i server that the source table is on.

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

Recommendation: Whenever you are replicating data from one System i table to another System i 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 greatly increases 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 System i 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 System i 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 a System i table that uses remote journaling, make sure that your remote journal is in an active state.

Restrictions: The following restrictions apply to registered tables that use remote journaling:

- Replica target table types are not supported in a remote journal configuration.
- The query option SQL_FAST_DELETE_ROW_COUNT (also known as fast delete) causes journaling to end and should not be used for registered tables. To avoid fast delete you could use a WHERE clause in the delete, or you could set the SQL_FAST_DELETE_ROW_COUNT parameter in QAQQINI to none. Fast delete does not log the individual deletes.
- Do not reorganize the source table by using RGZPFM with ALWCANCEL *YES. RGZPFM with ALWCANCEL *YES will create a CE journal entry, which causes the Capture program to signal a full refresh. Use RGZPFM with ALWCANCEL *NO to reorganize a replication source table.

For more information about the remote journal function, see "Remote journal management" in the i5/OS Information Center.

Referential integrity on the target table when the source is System i

When you are replicating data from source tables on System i, you can have referential integrity (RI) constraints on the target tables but some restrictions apply.

Replication supports RI constraints on the target when the source is on System i and if the following conditions are met:

- The RI constraints at the source and target tables match.
- The source application uses explicit commitment control.
- The source application has a commit between the changes that are made to the parent and child tables. That is, the parent transaction is separate from that of its children.

- The parent and child tables are journalled to the same journal.
- The parent and child tables are in the same subscription set.
- The value of COMMIT_COUNT in the IBMSNAP_SUBS_SET table is not null for the set, which forces the Apply program to process the changes in a transactional order.
- Full refresh is done by dropping RI constraints on the target side during a time when the application is quiesced. Then the RI is put back when the tables are in synch. A test change is made and replicated before releasing the application.

Other factors might influence replication, such as whether a parent table exists that is not part of replication. By default. the Apply program processes the tables in a subscription set by table order and commits only after all the tables in the set have been processed.

For user copy targets, it is preferred that you have RI constraints on the source but not on the target and that all of the related tables are in the same subscription set. A set is treated as a single unit so that if an error occurs in processing of any one table the entire set fails. As a result, the subscription set could provide acceptable results.

Using relative record numbers (RRN) instead of primary keys (System i)

If you are registering a System i 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 by using the relative record numbers (RRN).

When you choose to replicate by 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 System i 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 by 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 3 on page 57 shows the various combinations of underlying tables and what type of source view and CD view results from each combination.

Table 3. Combinations of underlying tables for views

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

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.
- System i 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 *objowner* (object owner) and *objname* (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**. .
System i `ADDDPRREG` system command	Use the ADDDPRREG command to register a view on System i.

Maintaining CCD tables as sources

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	• 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	• 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	• Set CCD_OLD_SYNCHPOINT to a value that represents the minimum value of IBMSNAP_COMMITSEQ from the CCD table. • Set SYNCHPOINT to a value that represents the maximum value of IBMSNAP_COMMITSEQ from the CCD table. • Set SYNCHTIME to a value that represents the maximum timestamp value of IBMSNAP_LOGMARKER from the CCD table.

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

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 System i, 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	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.
	
	The following commands set the environment and create a subscription set named SET00 with an Apply qualifier of AQ00.
	`SET SERVER CAPTURE TO DB SAMPLE;`
	`SET SERVER CONTROL TO DB TARGET;`
	`SET OUTPUT CAPTURE SCRIPT "capsubset.sql"`
	`CONTROLSCRIPT "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";`

Replication Center	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**.

System i
 	
ADDDPRSUB system command
	Use the Add DPR subscription set (ADDDPRSUB) command to create a subscription set with either one member or no members.
	
	For example, 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.

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 populate the Apply control tables correctly and so that every Apply program connects to the correct server by 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 populate the Capture and Apply control tables correctly and so that every Apply program connects to the correct server by 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 populate the Apply control tables correctly and so that every Apply program connects to the correct server by 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_SYNCH_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_SYNCH_MINUTES value is NULL, or is set to a numeric value less than 1, the Apply program processes all eligible data for that set in a single Apply cycle. If your CD and UOW tables contain large volumes of data, this situation can lead to such problems as the database transaction log becoming full or a spill file overflowing. You can change MAX_SYNCH_MINUTES to a non-NULL value by using the following guidelines:

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

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

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

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

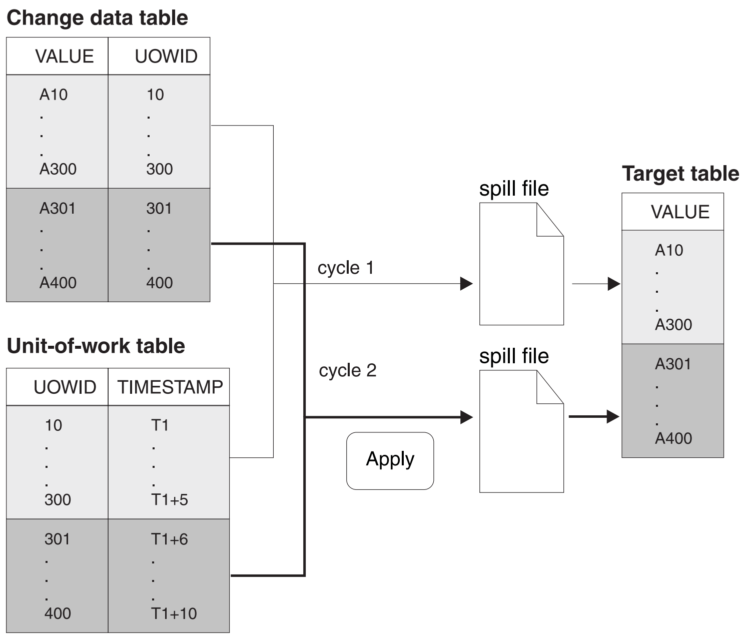


Figure 5. Data blocking. You can reduce the amount of network traffic by specifying a data-blocking value.

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 by 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 is typically much faster when you use 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, point-in-time, and CCD target tables. 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 by 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 4.

Table 4. 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 by using the clock at the Apply control server, and set the END_OF_PERIOD value by using the clock at the source server. This distinction is important if the two servers are in different time zones.

In Table 4, 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 by 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 processes this subscription set by using a regular time interval.
- Event-based replication

  The Apply program processes this subscription set whenever an event occurs.
- Both time-based and event-based replication

  The Apply program process this subscription set by using both a regular time interval and whenever an event occurs. In this case, the subscription set is 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.
- **z/OS:** Do not select ROWID columns for replication except when the ROWID column is the only unique index that is specified for replication.

  Recommendation: Use an IDENTITY column rather than a ROWID column as the unique index for replication.
- z/OS Linux UNIX Windows You can define a maximum of 200 members for each subscription set.
- System i 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	Use the CREATE MEMBER command to add a subscription-set member to an existing subscription set. For example, the following commands:
	• Set the environment.
	• Create a profile, TBSPROFILE, to store options for the table space 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.
	`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;`

Replication Center	Use one of the following notebooks: • 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**.

System i
`ADDDPRSUBM` system command
	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:
	`ADDDPRSUBM APYQUAL(AQHR) SETNAME(SETHR) SRCTBL(HR/YTDTAX)`
	`TGTTBL(TGTHR/TGTTAX)`

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.
- System i For source tables on System i 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
- z/OS 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 by 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 can 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 can 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:

Consistent-change-data (CCD) target tables provide committed transactional data that can be read and used by other applications, for example InfoSphere® DataStage®. You might also use a CCD table to audit the source data or keep a history of how the data is used.

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

Important for Version 10.1 on Linux, UNIX, and Windows: If the source database is DB2 10.1 for Linux, UNIX, and Windows with multiple DB2 pureScale® members, CCD targets are supported only if both the Capture and Apply programs are at Version 10.1 and the Capture **`compatibility`** parameter is set to 1001. With only a single DB2 pureScale member and **`compatibility`** set to 0801, CCD targets are supported even if the Apply program is at a version lower than 10.1.

By definition, a CCD table always includes the following columns in addition to the replicated columns from the source table:

Column	Description
IBMSNAP_INTENTSEQ	**Data type:** VARCHAR(16) FOR BIT DATA; **Nullable:** No A sequence number that uniquely identifies a change. This value is ascending in a transaction. z/OS The log sequence number (LRSN or RBA) of each update, delete, and insert.
IBMSNAP_OPERATION	**Data type:** CHAR(1); **Nullable:** No A flag that indicates the type of operation: I (INSERT), U (UPDATE), or D (DELETE).
IBMSNAP_COMMITSEQ	**Data type:** VARCHAR(16) FOR BIT DATA; **Nullable:** No A sequence number for each row within a transaction. z/OS The log sequence number (LRSN or RBA) of the source commit record.
IBMSNAP_LOGMARKER	**Data type:** TIMESTAMP; **Nullable:** No The approximate time that the data was committed.

When you create a noncomplete (COMPLETE=N) CCD table with the ASNCLP command-line program or Replication Center, you can specify additional auditing columns. The following table describes these columns:

Column	Description
IBMSNAP_AUTHID	**Data type:** VARCHAR(30); **Nullable:** Yes The user ID that updated the source table. z/OS This column is the primary authorization ID.
IBMSNAP_AUTHTKN	**Data type:** VARCHAR(30); **Nullable:** Yes The authorization token that is associated with the transaction. z/OS The correlation ID (normally a job name) that ran the source update.
IBMSNAP_PLANID	**Data type:** VARCHAR(8); **Nullable:** Yes z/OS The plan name that is associated with the transaction. This column will be null for DB2 for Linux, UNIX, and Windows.

Column	Description
IBMSNAP_UOWID	**Data type:** VARCHAR(16) FOR BIT DATA; **Nullable:** Yes The unit-of-work (UOW) identifier from the log record for a row. z/OS The unit-of-work identifier, sometimes called the unit-of-recovery ID (URID) of the transaction.

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

The middle tier in multi-tier configurations must be a DB2 table.

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 move the overhead of distribution from the source system to a second system. You can also avoid many of the database connections to your source system, thus moving the connection cost to the second tier. Also, 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).

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.

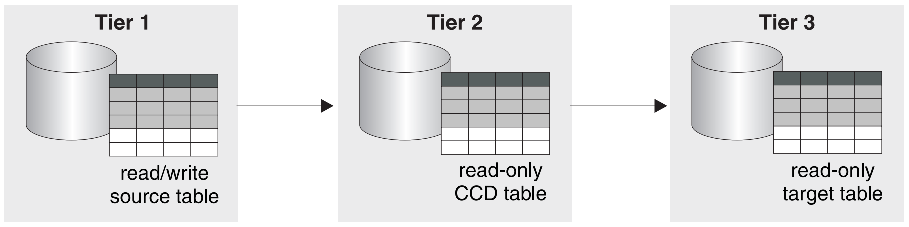


Figure 6. Three-tier replication model. You can replicate data from a source table to a target table, and then from that table to another target table.

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 the source as an external target table and register the table so that 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 with the same Capture schema.

   You can replicate to an external CCD table without joining the change-data (CD) table and the IBMSNAP_UOW table. The new table type is specified with a value of 9 in the TARGET_STRUCTURE column of the IBMSNAP_SUBS_MEMBR table. Although the type 9 CCD table includes the IBMSNAP_LOGMARKER column, the Apply program does not require a join of the CD table and IBMSNAP_UOW table to obtain the source commit timestamp for this column. Instead, the Apply program generates the same value in the IBMSNAP_LOGMARKER column for all of the rows in the same cycle.

The new CCD table type has the same structure as a type 3 CCD table. The table contains four mandatory IBM® columns in addition to the user columns:

```
IBMSNAP_COMMITSEQ
IBMSNAP_INTENTSEQ
IBMSNAP_OPERATION
IBMSNAP_LOGMARKER
user_columns
```

This target table type can be registered as a source table for a three-tier replication configuration.

Important: For type 9 CCD tables, the data blocking factor (MAX_SYNCH_MINUTES in the IBMSNAP_SUBS_SET control table) should be unset (`NULL`).

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.

  Note: With a condensed and complete CCD source table, if the same row is updated multiple times within an Apply cycle, an extra, duplicate row with the oldest key value might remain in the target table. This situation can occur because in condensed CCD sources, only the last update of the multiple key updates remains in the source CCD table. The Apply program cannot tell what the original value was and thus cannot remove the row with the oldest key value from the target table.

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.

Note: 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 by 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 i5/OS®.

Using the Replication Center

When you are creating a target table by 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 5. Mapping a DB2 DATE column to an Oracle DATE column

DB2 Column	Nickname Data Mapping	Oracle Column
A_DATE DATE	A_DATE TIMESTAMP A_DATE DATE	A_DATE DATE

The Oracle target table is created with an Oracle data type of DATE (which can contain both date and timestamp data). The initial nickname for an Oracle DATE data type in a federated database maps the DB2 data type as a TIMESTAMP. The DB2 Replication Center and the System i 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 by using the constraint columns.

  System i For a System i 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.

  System i For target tables on System i that use the RRN as the target key, you should run the Apply program on System i 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 by using before-image values, the Apply program is able to find the row with the old key value, and update the row by 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 SQL 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 or higher with APAR PK13542 or higher.

Encrypt scalar function in DB2 for Linux, UNIX, and Windows

Column data can be encrypted and decrypted by using the encrypt scalar function 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.

z/OS **FIELDPROC**

SQL Replication supports columns that are defined on DB2 for z/OS tables with field procedures (FIELDPROC) to transform values. The DB2 subsystem that contains the tables with FIELDPROC columns must be at APAR PK75340 or higher.

If possible, you should create the following index on your SYSIBM.SYSFIELDS table to improve performance:

```
CREATE INDEX "SYSIBM"."FIELDSX"
ON "SYSIBM"."SYSFIELDS"
(TBCREATOR ASC,
TBNAME ASC,
NAME ASC)
USING STOGROUP SYSDEFLT PRIQTY 100 SECQTY 100
CLOSE NO;
COMMIT;
```

Data type restrictions

SQL Replication cannot replicate the following data types:

- DB2 XML
- LOB columns from non-DB2 relational sources
- Any column on which a VALIDPROC is defined.

- You can replicate BINARY or VARBINARY data types when the source and target are on z/OS. Replication of these data types from a z/OS source to a DB2 for Linux, UNIX, and Windows target or federated target is not supported. BINARY and VARBINARY data types are supported as targets of source expressions only if the source datatype is CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, or ROWID.

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 and LONG VARGRAPHIC) 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. When you specify DATA CAPTURE CHANGES for a source table when the table is created, any LONG VARCHAR and LONG VARGRAPHIC columns are automatically enabled for replication. If you add LONG VARCHAR columns to the table after it is registered as a source and the table previously had no LONG columns, you must use the ALTER TABLE statement to enable DATA CAPTURE CHANGES INCLUDE LONGVAR COLUMNS for the new LONG VARCHAR or LONG VARGRAPHIC columns.

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 15 LOB columns or fewer per table. If you register a table with more than 15 LOB columns, the Apply program returns an error message. The Replication Center returns an error message if you attempt to register more than 15 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 by 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 System i.
- When you use LOBs in update-anywhere replication, you must set the conflict level to 0.

Replication of new DB2 Version 9.7 data types (Linux, UNIX, Windows)

SQL Replication supports new data types that were introduced with DB2 for Linux, UNIX, and Windows Version 9.7 to make it easier to migrate applications to DB2.

Some of the new data types require special considerations for a replication environment. The following sections provide details:

- “TIMESTAMP with extended precision”
- “DATE with compatibility option” on page 94
- “NUMBER” on page 94

TIMESTAMP with extended precision

SQL replication supports replication of TIMESTAMP data with extended precision that ranges from TIMESTAMP(0) to TIMESTAMP(12). You can map columns of non-matching precision. If both the source and target databases and Capture and Apply are Version 9.7 or newer, the source data is padded or truncated at the target database.

In a mixed-level environment where only the source DB2 is at Version 9.7, TIMESTAMP columns might also require padding or truncation. Replication of such columns can occur only when both Capture and Apply are at Version 9.7 or later. For example, if you replicated a source at V9.7 to a target at V9.5 and a registered table included a TIMESTAMP(12) column, the V9.7 Apply would truncate six-digits from the fractional seconds portion of the TIMESTAMP value. The truncation is necessary because DB2 Version 9.5 does not support extended precision, and so for V9.5 databases TIMESTAMP values have a fractional seconds portion that equates to the default V9.7 precision of TIMESTAMP(6). Table 6 on page 94 shows the value at the source and resulting truncated value at the target.

z/OS **Note:** When handling these new data types, SQL replication treats a DB2 for z/OS source or target the same as DB2 for Linux, UNIX, and Windows Version 9.5 or older.

Table 6. Truncation of TIMESTAMP(12) during replication

Source value in TIMESTAMP(12)	Target value in TIMESTAMP(6)
2009-07-10-10.33.42.458499823012	2009-07-10-10.33.42.458499

If the target database is older than V9.7, TIMESTAMP values of lower precision than the default TIMESTAMP(6) are padded automatically by DB2 so the fractional seconds portion contains six places.

DATE with compatibility option

The date compatibility option stores the DATE type with an additional time portion (HH:MM:SS). This format conforms to the date representation by other relational database management systems such as Oracle, where the DATE data type includes YYYY-MM-DD HH:MM:SS.

SQL Replication treats databases without date compatibility the same as DB2 databases prior to V9.7, and the same as DB2 for z/OS subsystems. When date compatibility is enabled, DB2 handles columns that are defined as DATE in the same way that it handles columns defined as TIMESTAMP(0).

Enable the DATE as TIMESTAMP(0) support by setting bit position number 7 (0x40) of the DB2_COMPATIBILITY_VECTOR registry variable before you create a database. With SQL Replication you can create the following column mappings between DATE and TIMESTAMP(0):

DATE to TIMESTAMP(0)
If the source database does not have date compatibility enabled, the target value is padded to YYYY-MM-DD-00:00:00.

TIMESTAMP(0) to DATE
If the target database does not have date compatibility enabled, the TIMESTAMP(0) value is truncated to YYYY-MM-DD.

NUMBER

The NUMBER data type supports applications that use the Oracle NUMBER data type. DB2 treats NUMBER data internally as DECFLOAT if no precision or scale are specified, and as DECIMAL with precision or scale if these attributes are specified.

Because SQL Replication already supports DECFLOAT and DECIMAL, you can map columns defined with any of these three numeric types to each other: NUMBER to DECFLOAT or DECIMAL, DECFLOAT to NUMBER or DECIMAL, and DECIMAL to NUMBER or DECFLOAT.

Replication of tables with identity columns

SQL Replication allows identity columns in both source and target tables, but because of DB2 restrictions you might need to take extra steps if your source table has columns that are defined with the AS IDENTITY GENERATED ALWAYS clause.

Identity columns are handled differently by replication depending on whether they are in the source or target table:

Source table
If you have an identity column in a source table and you want to replicate it to a target table, register and subscribe to the source table as usual. The CD and target tables are created with numeric columns to hold the values. For example, a source column that is defined as GENERATE ALWAYS might be replicated to a BIGINT column at the target. The columns in the CD and target table cannot be identity columns themselves, so you cannot replicate an identity column in a source table to an identity column in a target table.

Target table
If you have an identity column in a target table, do not include that column in your replication configuration when defining the subscription-set member. The column is populated automatically when replication inserts into or updates the target table. The behavior of the identity column is the same as for inserts and updates by any other application. If you replicate the same source table to multiple target tables that have identity columns, the identity values in those target tables are independent of each another.

DB2 does not allow inserts into columns that are defined with the AS IDENTITY GENERATED ALWAYS clause, and so this clause is not supported for SQL Replication target tables. However, options exist for replicating these columns:

- Create the target table without the IDENTITY clause.
- Create the target table with a column that is defined with AS IDENTITY GENERATED BY DEFAULT.

For columns that are defined with AS IDENTITY GENERATED BY DEFAULT, the range of values must be distinct between the source and the target because DB2 does not guarantee uniqueness of identity columns between two different DB2 databases.

For example, the identity column at one site could be set to even numbers (START WITH 2, INCREMENT BY 2) and at the other site the identity column could be set to odd numbers (START WITH 1, INCREMENT BY 2). You could also assign ranges to sites (for example, 1 to 10,000 at one site and 20,000 to 40,000 at the other). The odd-even approach ensures that in a conflict situation, two different rows that accidentally have the same generated identity key do not overwrite one another when the conflict action is to force the change.

The data type of the identity column (SMALLINT, INTEGER, or BIGINT) should be determined by application needs, for example the largest number that you expect in the column.

The identity columns should be NO CYCLE if numbers cannot be reused. Put a plan in place for what to do when the maximum value is reached (SQLSTATE 23522). If you use CYCLE, make sure that a new use of a number does not cause problems for any existing use of the number, including what happens during replication.

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 subset data at subscription.

Views are used to subset data at registration time, while query predicates are used to subset data at subscription time. In many situations, it depends on your preference whether to use subscription predicates or registered views. A few factors might influence you:

- Views might already exist and meet the qualifications to be a registered view for replication.
- You might find views to be an easier approach to verify the subsetting that you defined for replication.
- Subscription predicates are stored in replication control tables, which eliminates the need to create and manage views.

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.

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.

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 does not let you prevent certain changes in those rows from being replicated. In some replication scenarios, you might want to prevent certain changes in rows from being captured and replicated to the target tables. For example, if you want your target tables to contain all rows and you never want any rows deleted from them, you do not want to replicate 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 by 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 gives 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 102
- "Creating computed columns" on page 103

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 that uses 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 by using stored procedures or SQL statements

When you define subscription set information, you can also define run-time processing statements by 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.

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 (which use 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 System i, 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 use the Replication Center or ASNCLP command-line program to define a subscription-set member and the target table that is 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.

Before you begin

When you create expressions that reference source table columns, prefix the source column name with a colon (:) and after the column name add a space. For example, `:COL1 `.

Using SQL expressions, you can 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 by 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 by 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 the result so that it is applied to the target column TGT_COL2.

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.

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

Note: 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**.
z/OS Linux UNIX Windows **`asncap`** system command	Use this command to start the Capture program and optionally specify startup parameters.

Method	Description

z/OS
 	
z/OS console or TSO
	On z/OS, you can start the Capture program by using JCL or as a system-started task. You can specify new invocation parameter values when you start a Capture program with JCL.
	
	z/OS has a 100-byte limit for the total length of parameters that you can specify in the PARMS= field. To overcome this limitation, replication programs now allow you to specify as many additional parameters as needed in the SYSIN data set.
	
	When the SYSIN DD statement is included in the invocation JCL, the Capture program automatically concatenates what is specified in the SYSIN dataset to the PARMS= parameters. You can only specify Capture parameters in the SYSIN data set. Any LE parameters must be specified in the PARMS= field or in LE _CEE_ENVFILE=DD, followed by a slash(/).
	
	Example:
	`//* asterisk indicates a comment line`
	`// CAP EXEC PGM=ASNCAP,PARMS='LE/Capture parameters'`
	`//* Parameters can be any or no LE parameters and any or`
	`//* no Capture parameters`
	`//SYSIN DD *`
	`//* additional Capture parameters, one or more`
	`//* parameters on each line`
	`  CAPTURE_SERVER=DSN!! CAPTURE_SCHEMA=CAPCAT`
	`  DEBUG=Y LOGSTDOUT=N`

Windows 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 **`asnccmd 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 from a known point in the DB2 log

You can prompt the Capture program to reread the DB2 recovery log from a known point and reprocess log records that were already captured and applied.

About this task

Important: This procedure should only be used when the target table is a user copy.

Procedure

1. Stop the Capture and Apply programs.
2. Set the Capture RETENTION_LIMIT and LAG_LIMIT values to their maximum, as shown in the following SQL statement:
   ```
   UPDATE ASN.IBMSNAP_CAPPARMS SET RETENTION_LIMIT=99999,LAG_LIMIT=99999;
   ```
3. If the SYNCHPOINT values in the IBMSNAP_UOW, CD, IBMSNAP_REGISTER, and IBMSNAP_PRUNCNTL tables are higher than the LSN value from which you want to start Capture, use SQL to set the value to the point from which you want to start recapturing transactions. In the following example, 00000006F5638E60000 is the log sequence number and 2009-09-05-09.55.43.316970 is the timestamp from which you send the Capture program back to start reading the log.
   ```
   UPDATE ASN.IBMSNAP_REGISTER SET SYNCHPOINT = x'00000006F5638E600000',
   SYNCHTIME=TIMESTAMP('2009-05-05-09.55.43.316970');

   UPDATE ASN.IBMSNAP_REGISTER SET CD_OLD_SYNCHPOINT=x'00000006F5638E600000',
   CD_NEW_SYNCHPOINT=x'00000006F5638E600000',
   CCD_OLD_SYNCHPOINT=x'00000006F5638E600000'
   WHERE GLOBAL_RECORD='N';

   UPDATE ASN.IBMSNAP_SUBS_SET SET
   LASTRUN=TIMESTAMP('2009-09-05-09.55.43.316970'),
   LASTSUCCESS=TIMESTAMP('2009-05-05-09.55.43.316970'),
   SYNCHPOINT=x'00000006F5638E600000',
   SYNCHTIME=TIMESTAMP('2009-05-05-09.55.43.316970')
   WHERE WHOS_ON_FIRST='S' AND SET_NAME='BACK1';

   UPDATE ASN.IBMSNAP_PRUNCNTL SET SYNCHPOINT =x'00000006F5638E600000',
   SYNCHTIME=TIMESTAMP('2009-05-05-09.55.43.316970');

   UPDATE ASN.IBMSNAP_PRUNE_SET SET SYNCHPOINT =x'00000006F5638E600000',
   SYNCHTIME=TIMESTAMP('2009-05-05-09.55.43.316970');

   DELETE FROM ASN.IBMSNAP_UOW;


   INSERT INTO ASN.IBMSNAP_RESTART (MAX_COMMITSEQ, MIN_INFLIGHTSEQ,
   MAX_COMMIT_TIME,CURR_COMMIT_TIME,CAPTURE_FIRST_SEQ)
   values (,x'00000006F5638E600000',
   '2009-05-05-09.55.43.316970','2009-05-05-09.55.43.316970',
   x'00000006F5638E600000');
   ```
4. Start the Capture program in WARMNS mode, and start the Apply program with your typical startup parameters.

Starting the Capture program (System i)

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 System i, use one of the following methods:

Method	Description
`STRDPRCAP` system command (System i)	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 7 and Table 8 on page 109.

Linux UNIX Windows

Table 7. 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
`asynchlogrd`	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

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

Operational parameter	Default value	Column name in IBMSNAP_CAPPARMS table
monitor_interval	300 seconds	MONITOR_INTERVAL
memory_limit	32 MB	MEMORY_LIMIT
autoprune	y[3]	AUTOPRUNE
term	y[3]	TERM
autostop	n[4]	AUTOSTOP
caf	n[4]	not applicable
logread_prefetch	Y [3] for partitioned databases; N [4] for nonpartitioned databases	not applicable
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

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.

System i

Table 8. Default settings for Capture operational parameters (System i)

Operational parameter	Default value	Column name in IBMSNAP_CAPPARMS table
CAPCTLLIB	ASN[1]	not applicable
JOBD	*LIBL/QZSNDPR	not applicable
JRN	*ALL	not applicable
RETAIN	10080 minutes	RETENTION_LIMIT
LAG	10080 minutes	LAG_LIMIT
FRCFRQ	30 seconds	COMMIT_INTERVAL
CLNUPITV	*IMMED [2]	not applicable
CLNUPITV	86400 seconds[2]	PRUNE_INTERVAL
CLNUPITV	*IMMED [2]	not applicable
TRCLMT	10080 minutes	TRACE_LIMIT
MONLMT	10080 minutes	MONITOR_LIMIT

Table 8. Default settings for Capture operational parameters (System i) (continued)

Operational parameter	Default value	Column name in IBMSNAP_CAPPARMS table
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.

All parameters apply to z/OS, Linux, UNIX, and Windows, unless otherwise noted.

- "add_partition (Linux, UNIX, Windows)" on page 111
- "asynchlogrd" on page 111
- "autoprune" on page 111
- "autostop" on page 111
- "caf (z/OS)" on page 112
- "capture_path" on page 112
- "capture_schema" on page 113
- "capture_server" on page 113
- "commit_interval" on page 114
- "hs (z/OS)" on page 114
- "ignore_transid" on page 114
- "lag_limit" on page 115
- "logreuse" on page 115
- "logstdout" on page 116
- "memory_limit" on page 116
- "monitor_interval" on page 116
- "prune_interval" on page 117
- "retention_limit" on page 117
- "sleep_interval" on page 118
- "startmode" on page 118
- "term" on page 119
- "trace_limit" on page 119

Linux UNIX Windows

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.

asynchlogrd

Default: **asynchlogrd**=n

The **asynchlogrd** parameter specifies that you want the Capture program to use a dedicated thread for capturing transactions from the DB2 recovery log. The transaction reader thread prefetches committed transactions in a memory buffer, from which another thread gets the transactions and processes them into SQL statements for insertion into the CD table. This asynchronous mode can improve Capture throughput in all environments with particular benefits for partitioned databases and z/OS data-sharing.

On systems with very high activity levels, this prefetching might lead to more memory usage. Adjust the **memory_limit** parameter accordingly. If you have a low volume of changes, you might prefer the default value of N to reduce CPU consumption.

autoprune

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

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 overhead to the administration of replication (for example, you need to keep restarting the Capture program).

z/OS

caf (z/OS)

Default: n

The default option is **caf** =n. You can override this default and prompt the Capture program to use the Call Attach Facility (CAF) by specifying the **caf** =y option. The **caf** =y option specifies that the replication program overrides the default Recoverable Resource Manager Services (RRS) connect and runs with CAF connect.

If RRS is not available you will get a message and the replication program switches to CAF. The message warns that the program was not able to initialize a connection because RRS is not started. The program attempts to use CAF instead. The program runs correctly with CAF connect.

capture_path

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.

z/OS

Because the Capture program is a POSIX application, the default Capture path depends on how you start the program:

USS command prompt
The directory where you started the program.

Started task or through JCL
The home directory in the USS file system of the user ID that is associated with the started task or job.

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 that are stored in an explicitly named directory path, with the HLQ concatenated as the first part of the file name. For example, `sysadm.CAPV8.`*`filename`*. Using an HLQ might be convenient if you want to have the Capture log and LOADMSG files be system-managed (SMS).

If you want the Capture started task to write to a .log data set with a user ID other than the ID that is executing the task (for example TSOUSER), you must specify a single quotation mark (‘) as an escape character when using the SYSIN format for input parameters to the started task. For example, if you wanted to use the high-level qualifier JOESMITH, then the user ID TSOUSER that is running the Capture program must have RACF authority to write data sets by using the high-level qualifier JOESMITH, as in the following example:

```
//SYSIN    DD  *
 CAPTURE_PATH=//'JOESMITH
/*
```

Linux UNIX Windows

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`. If you start the Capture program as a Windows service, by default the Capture program starts in the `\sqllib\bin` directory.

capture_schema

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 with 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 by using the other schema.

capture_server

z/OS **Default: capture_server**=None

Linux UNIX Windows **Default:** **`capture_server`**=value of DB2DBDFT environment variable, if it is set

The **`capture_server`** parameter specifies the Capture control server.

z/OS You must specify the **`capture_server`** parameter. 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.

Linux UNIX Windows 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.

commit_interval

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.

hs (z/OS)

Default: **`hs`**=n

The **`hs`** parameter specifies whether the Capture program creates one or more spill files in hiperspace (high performance data space) if Capture exceeds its memory limit during an attempt to write a row in memory. By default (**`hs`**=n) Capture creates the spill file on disk or virtual input/output (VIO).

Recommendation: Allocate enough memory to the Capture job to avoid the need for spill files.

ignore_transid

Default: None

The **`ignore_transid`**=*transaction_ID* parameter specifies that the Capture program ignores the transaction that is identified by *transaction_ID*. The transactions are not replicated or published. You can use this parameter if you want to ignore a very large transaction that does not need to be replicated, for example a large batch job. The value for *transaction_ID* is a 10-byte hexadecimal identifier in the following format:

z/OS

0000:*xxxx*:*xxxx*:*xxxx*:*mmmm*

Where *xxxx*:*xxxx*:*xxxx* is the transaction ID, and *mmmm* is the data-sharing member ID. You can find the member ID in the last 2 bytes of the log record header in the LOGP output. The member ID is 0000 if data-sharing is not enabled.

Linux UNIX Windows

nnnn:0000:*xxxx*:*xxxx*:*xxxx*

Where *xxxx*:*xxxx*:*xxxx* is the transaction ID, and *nnnn* is the partition identifier for partitioned databases (this value is 0000 if for non-partitioned databases).

Tip: The shortened version **transid** is also acceptable for this parameter.

lag_limit

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

Default: **logreuse**=n

The Capture program stores operational information in a log file.

z/OS The log file name 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.

Linux UNIX Windows The name of the log file is *db2instance*.*capture_server*.*capture_schema*.CAP.log. For example, DB2INST.SRCDB1.ASN.CAP.log.

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

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

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

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

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

Default: **prune_interval**=300

The **prune_interval** parameter specifies how often the Capture program tries to prune old rows from some of its control tables. This parameter is valid only if **autoprune**=y.

By default, the Capture program prunes the CD and UOW tables every 300 seconds (five minutes). If the tables are not pruned often enough, the table space that they are in can run out of space, which forces the Capture program to stop. If they are pruned too often or during peak times, the pruning can interfere with application programs running on the same system. You can set the optimal pruning frequency for your replication environment. Performance will generally be best when the tables are kept small.

Before you lower the prune interval, ensure that data is being applied frequently so that pruning can occur. If the Apply program is not applying data frequently, it is useless to set the prune interval lower because the Apply program must replicate the data to all targets before the CD and UOW tables can be pruned.

The prune interval determines how often the Capture program tries to prune the tables. It works in conjunction with the following parameters, which determine when data is old enough to prune: **trace_limit**, **monitor_limit**, **retention_limit**. For example, if the **prune_interval** is 300 seconds and the **trace_limit** is 10080 seconds, the Capture program will try to prune every 300 seconds. If it finds any rows in the trace table that are older than 10080 minutes (7 days), it will prune them.

retention_limit

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

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

Default: **`startmode`**=warmsi

You can start Capture by 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

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

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.

Linux UNIX Windows

Example 1

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 by 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 with a 20-second commit interval, you monitor its performance.
   ```
   asncap capture_server=srcdb1 capture_schema=asnprod commit_interval=20
   ```
3. You decide that you want to try an even lower commit interval. Instead of stopping the Capture program, you submit a change parameters request that sets the commit interval to 15 seconds. The Capture program continues to run, only now it commits data every 15 seconds.
   ```
   asnccmd capture_server=srcdb1 capture_schema=asnprod chgparms
   commit_interval=15
   ```

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

System i

Example 2

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 by 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 with 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 System i, you cannot 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

z/OS Linux UNIX Windows You can change the following Capture parameters while the Capture program is running:

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

Restriction: z/OS The amount of memory that the Capture program can use to build messages is determined when the Capture program starts, based on the value of the **memory_limit** parameter and the REGION size that is specified in the JCL. The value of **memory_limit** cannot be altered with the Capture program is running. To change the value you must first stop the Capture program.

System i 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**.
z/OS Linux UNIX Windows **asnccmd chgparms** system command	This method does not show the current values of the parameters.
System i **OVRDPRCAPA** system command	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 by using startup parameters or while the program is running.

About this task

Only one row is allowed in the IBMSNAP_CAPPARMS table, and the row is required. If you want to change one or more of the default values, you can update columns instead of inserting rows.

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**.
System i **`CHGDPRCAPA`** system command	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

System i 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**.
z/OS Linux UNIX Windows **`asnccmd stop`** system command	Use this command to stop Capture.
System i **`ENDDPRCAP`** system command	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 **autoprune** 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 System i, 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**.
z/OS Linux UNIX Windows **asnccmd reinit** system command	Use this command to reinitialize Capture.
System i **INZDPRCAP** system command	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**.
z/OS Linux UNIX Windows **`asnccmd 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 **`autoprune`** parameter.

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

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

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**.
z/OS Linux UNIX Windows **`asnccmd 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

Operating the Apply program includes such tasks as starting and stopping and using the ASNDONE and ASNLOAD exit routines.

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.
- z/OS You manually bound the Apply program to all necessary servers.
- Linux UNIX Windows A password file exists for end-user authentication for remote servers.

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**.
z/OS Linux UNIX Windows **`asnapply`** system command	Use this command to start Apply.

z/OS
 	
z/OS console or TSO
	On z/OS, you can start the Apply program by using JCL or as a system-started task. You can specify new invocation parameter values when you start an Apply program with JCL.
	
	z/OS has a 100-byte limit for the total length of parameters that you can specify in the PARMS= field. To overcome this limitation, replication programs now allow you to specify as many additional parameters as needed in the SYSIN data set.
	
	When the SYSIN DD statement is included in the invocation JCL, the Apply program automatically concatenates what is specified in the SYSIN data set to the PARMS= parameters. You can only specify Apply parameters in the SYSIN data set. Any LE parameters must be specified in the PARMS= field or in LE _CEE_ENVFILE=DD, followed by a slash(/).
	
	Example:
	`//* asterisk indicates a comment line`
	`// APP EXEC PGM=ASNAPP,PARMS='LE/Apply parameters'`
	`//* Parameters can be any or no LE parameters and any or`
	`//* no Apply parameters`
	`//SYSIN DD *`
	`//* additional Apply parameters, one or more`
	`//* parameters on each line`
	`  APPLY_SERVER=DSN!! APPLY_SCHEMA=APPCAT`
	`  DEBUG=Y LOGSTDOUT=N`

Windows 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 by 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 (System i)

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 with the COPYONCE(*YES) startup parameter.
- You specified ALWINACT(*NO) and there is no data to be processed.
- You stop the Apply program by 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 9 and Table 10 on page 131.

z/OS Linux UNIX Windows

Table 9. Default settings for Apply operational parameters (z/OS, Linux, UNIX, Windows)

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
caf	y[5]	not applicable
control_server	DB2DBDFT[2]	not applicable
copyonce	n[3]	COPYONCE
db2_subsystem	No default[4]	not applicable
delay	6 seconds	DELAY
errwait	300 seconds	ERRWAIT
inamsg	y[5]	INAMSG
loadxit	n[3]	LOADXIT
logreuse	n[3]	LOGREUSE
logstdout	n[3]	LOGSTDOUT
notify	n[3]	NOTIFY
opt4one	n[3]	OPT4ONE
pwdfile	asnpwd.aut	not applicable
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.

System i

Table 10. Default settings for Apply operational parameters (System i)

Operational parameter	**Description of (****value***)**
USER (*CURRENT)	The user who signed on to the system.
JOBD (*LIBL/QZSNDPR)	Product library name / job description.
APYQUAL (*USER)	Current user name (from above).
CTLSVR (*LOCAL)	Local RDB server name.
TRACE (*NONE)	Do not generate a trace.
FULLREFPGM (*NONE)	Do not run the ASNLOAD exit routine.
SUBNFYPGM (*NONE)	Do not run the ASNDONE exit routine.
INACTMSG (*YES)	When the Apply program begins an inactive period, it generates message ASN1044, which describes how long the program will be inactive.
ALWINACT (*YES)	Sleep if there is nothing to process.
DELAY (6)	Wait 6 seconds after an Apply cycle before processing again.
RTYWAIT (300)	Wait 300 seconds before retrying a failed operation.
COPYONCE (*NO)	Do not terminate after completing one copy cycle, continue processing.
TRLREUSE (*NO)	Do not empty the 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.

These parameters apply to z/OS, Linux, UNIX, and Windows unless otherwise specified.

- "apply_path" on page 132
- "apply_qual" on page 133
- caf
- "control_server" on page 133
- "copyonce" on page 134
- "db2_subsystem (z/OS)" on page 134
- "delay" on page 134
- "errwait" on page 135
- "inamsg" on page 135
- "loadxit" on page 135
- "logreuse" on page 136
- "logstdout" on page 136
- "notify" on page 136
- "opt4one" on page 136
- "pwdfile" on page 137

- "sleep" on page 137
- "spillfile" on page 138
- "sqlerrcontinue" on page 138
- "term" on page 139
- "trlreuse" on page 139

apply_path

Linux UNIX Windows **Default:** **`apply_path`**=*current_directory*

Windows **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.

You can specify either a path name or a high-level Qualifier (HLQ), such as //APPV9. 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 the names that are stored in an explicitly named directory path, with the HLQ concatenated as the first part of the file name. For example, `sysadm.APPV9.filename`. Using an HLQ might be convenient if you want to have the Apply log and LOADMSG files be system-managed (SMS).

If you want the Apply started task to write to a .log data set with a user ID other than the ID that is executing the task (for example TSOUSER), you must specify a single quotation mark (‘) as an escape character when using the SYSIN format for input parameters to the started task. For example, if you wanted to use the high-level qualifier JOESMITH, then the user ID TSOUSER that is running the Apply program must have RACF authority to write data sets by using the high-level qualifier JOESMITH, as in the following example:

```
//SYSIN    DD  *
 APPLY_PATH=//'JOESMITH
/*
```

z/OS 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.

Windows **Starting instances of Apply on one Windows system**: When you start the Apply program by 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

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.

z/OS

caf

Default: y

The runtime parameter **caf** =y specifies whether the Apply program overrides the Recoverable Resource Manager Services (RRS) connect and runs with Call Attach Facility (CAF) connect. The **caf** =y option is the default for the Apply program.

control_server

z/OS **Default:** None

Linux UNIX Windows **Default:** The value of the DB2DBDFT environment variable, if available

The Apply control server is the server on which the Apply control tables and subscription definitions reside. Specify only one control server per Apply qualifier. If you do not specify a value, the Apply program starts on the default server. The default depends on your operating system.

z/OS On z/OS, you must specify the **control server** parameter.

If the Apply program cannot connect to the control server, it follows the action set by the **term** parameter:

term=y (default)
: The Apply program terminates.

term=n
: Apply waits for the amount of time set by the **errwait** parameter, then retries the connection.

The Apply worker thread sets its state to "waiting for database" if it cannot connect to its Apply control server and if the Apply program was started with the **term**=n parameter. You can run the status command in **asnacmd** or MODIFY on z/OS to check whether the Apply worker thread is running but unable to connect to the control server.

If the Apply program cannot connect to other servers, it issues an error message and continues processing.

copyonce

Default: **copyonce**=n

The **copyonce** parameter determines the copy cycle for the Apply program.

When you start the Apply program by 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 by 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**=ninstead 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

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

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 a 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

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

Default: **`loadxit`**=`n`

The **`loadxit`** parameter specifies whether or not the Apply program should refresh target tables by 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.

z/OS On z/OS, the ASNLOAD exit routine uses the DSNUTILS stored procedure to call DB2 utilities that are required to load the target table.

logreuse

Default: **logreuse**=n

The Apply program stores operational information in a log file. The parameter specifies whether to append to the log file or to overwrite it.

z/OS
The name of the log file is *control_server.apply_qualifier*.APP.log.

Linux UNIX Windows
The name of the log file is *db2instance.control_server.apply_qualifier*.APP.log.

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

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

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

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 by 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

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

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

z/OS **Default: `spillfile`**`=MEM`

Linux UNIX Windows **Default: `spillfile`**`=disk`

Apply retrieves data from the source tables and places it in a spill file on the system where the Apply program is running.

z/OS On z/OS, 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. You can also specify **`spillfile`**`=hs` and Apply will use high performance data space (hiperspace) for spilling.

Linux UNIX Windows The only valid setting for **`spillfile`** is disk because spill files are always on disk in the location specified by the **`apply_path`** parameter.

sqlerrcontinue

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 by 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 by 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):

23505/-803
  Duplicate index violation

You would create an SQL state file that contains the following SQL state:

```
23505
```

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

Default: **term**=`y`

The **`term`** parameter determines what the Apply program does if it cannot connect to its control server.

By default, the Apply program terminates if it cannot connect.

Use **`term`**=`n` if you want the Apply program to keep running. Apply logs an error, waits for the amount of time set by the **`errwait`** parameter, then retries the connection to its control server.

The **`term`** parameter is ignored if **`copyonce`** is specified.

trlreuse

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.

Linux UNIX Windows

Example

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 (z/OS, Linux, UNIX, Windows)

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 by 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 still starts 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 does 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**.
z/OS Linux UNIX Windows **`asnacmd stop`** system command	Use this command to stop Apply.
System i **`ENDDPRAPY`** system command	Use the End DPR Apply (ENDDPRAPY) command to stop an Apply program on your local system.

Where the Apply program stores details about referential integrity failures

You can find details about rows that were not applied to target tables because of referential integrity (RI) violations. The Apply program in SQL Replication writes these details to a file.

When you start the Apply program, it creates a file with the following name in the path that is specified by the **`apply_path`** parameter: *`apply_qualifier`*`.RI`. If the file exists when the Apply program starts, it clears the file and begins writing to the empty file if necessary.

If an INSERT, UPDATE, or DELETE operation on a target table fails with SQLCODE SQL0530, SQL0531, or SQL0532 because of an RI constraint, the Apply program writes the row information to the file.

Apply also writes the following information to the file for each row that received an error:
- Subscription set name
- SQLCA and SQLDA for the failing statement
- SQL statement that failed
- Values of the IBMSNAP_COMMITSEQ and IBMSNAP_INTENTSEQ columns from the CD table row that caused the error

When a row operation at the target table fails, the Apply writes the failing row to a separate retry file and continues to try the operation. Sometimes an RI violation can be resolved in this way and you do not need to take any action. If Apply continues to get RI errors for all the rows in the retry file, Apply issues the ASN0999E message and the current Apply cycle fails. In this situation, you might need to check the RI file to identify the cause of the errors.

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

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 by 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.
   - z/OS See the PROLOG section of the sample program SASNSAMP(ASNDONE).
   - 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.
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 (System i)

You can customize the ASNDONE exit routine on System i 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 can define run-time processing statements by 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 11 indicates where you can find the source code for this routine in C, COBOL, and RPG languages:

Table 11. Source code for ASNDONE

Compiler language	Library name	Source file name	Member name
C	QDP4	QCSRC	ASNDONE
COBOL	QDP4	QCBLLESRC	ASNDONE
RPG	QDP4	QRPGLESRC	ASNDONE

When modifying the program, consider these activation group concerns:

If the program is created to run with a new activation group
The Apply program and the ASNLOAD program will not share SQL resources, such as relational database connections and open cursors. The activation handling code in the System i 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 by 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 by 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 sample exit routine differs on each DB2 platform to take advantage of the utility options offered on that platform:

z/OS Linux UNIX Windows

The ASNLOAD exit routine is shipped as a sample exit routine in both a source format and a compiled format.

System i

ASNLOAD is shipped in a source format only.

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.

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 you use it.

Before you begin

- 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 by 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 by 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 by 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 are populated by 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 you use it.

Before you begin

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 with 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 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 by using the code that 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 (z/OS, Linux, UNIX, Windows)

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.

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.

Linux UNIX Windows 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.

Linux UNIX Windows **4**

Use a combination of the EXPORT utility and the LOAD utility.

Linux UNIX Windows **5**

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

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 (System i)

You can use the ASNLOAD exit routine to refresh target tables more efficiently on System i. 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 and specify 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 12.

Table 12. Source code for ASNLOAD

Compiler language	Library name	Source file name	Member name
C	QDP4	QCSRC	ASNLOAD
COBOL	QDP4	QCBLLESRC	ASNLOAD
RPG	QDP4	QRPGLESRC	ASNLOAD

2. Compile, link, and bind the program and place the executable in the appropriate directory. To avoid interference with the Apply program, compile the exit routine so that it uses a new activation group (not the activation group of the caller).

   You can compile the exit routine with a named activation group or with a new activation group. To get better performance, use a named activation group. With a named activation group, the exit routine must commit or roll back changes as needed. The Apply program will not cause changes to be committed or rolled back (unless it ends). The exit routine should either explicitly commit changes, or it should be compiled to implicitly commit changes when it completes. Any uncommitted changes when the exit routine completes are not committed until either:

   - The Apply program calls another exit routine with the same activation group.
   - The job started for the Apply program ends.
3. Start the Apply program with the FULLREFPGM parameter set to the name of the ASNLOAD program. When you start the Apply program, it uses the

ASNLOAD exit routine that you specified. If you want it to use another ASNLOAD exit routine, end the Apply program and start it again.

When you run the ASNLOAD exit routine, it refreshes all the target tables, table by table.

Refreshing one table in a multi-table subscription set

You can prompt the replication programs to reload one table in a subscription set that contains multiple tables. To do this, you use SQL to update values in the IBMSNAP_SUBS_MEMBR table.

To refresh a single table in a multi-table set, run the following SQL update statement at the Apply control server. The SQL resets the value of the MEMBER_STATE column to N, prompting a full refresh of the table.

```
UPDATE ASN.IBMSNAP_SUBS_MEMBR
SET MEMBER_STATE='N'
WHERE APPLY_QUAL= apply_qual AND
SET_NAME = set_name AND
WHOS_ON_FIRST = whos_on_first AND
SOURCE_OWNER = source_owner AND
SOURCE_TABLE = source_table AND
TARGET_OWNER = target_owner AND
TARGET_TABLE = target_table
```

Note: Any tables that you do not want to refresh should have a value of L or S in the MEMBER_STATE column.

In the next cycle, the Apply program performs a full refresh for any members with a value of N and changes the value of MEMBER_STATE to L after the target table is loaded. For subsequent cycles, differential refresh is resumed for all members in the subscription set from the synchpoint where the set was stopped.

Ensuring that utilities used for full refresh wait for committed data

If you specify that the Apply program use the ASNLOAD exit routine with a DB2 for Linux, UNIX, and Windows Version 9.7 and newer source, you must ensure that the utilities that are used for the full refresh wait for committed data.

About this task

Starting with Version 9.7, DB2 by default uses *currently committed* semantics, in which applications such as load utilities that read table data do not wait for writing applications to release row locks. Instead, reading applications return data that is based on the currently committed version; that is, data prior to the start of the write operation.

Because a full refresh of the target table requires the latest committed data from the source table, you must ensure that the utilities wait until all in-progress transactions that modify the source table are completed before beginning the load. This behavior is known as "wait for outcome."

If you specify the LOAD from CURSOR utility for ASNLOAD, you must use a new federated server option, CONCURRENT_ACCESS_RESOLUTION=W, to enforce wait for outcome behavior on the nickname that is used for the full refresh.

Restriction: You can only set this option for a registered server of type DB2/UDB Version 9.7 or newer.

Note these considerations:

- The procedures for setting concurrent access for the LOAD from CURSOR utility are different if the Apply control server is at Version 9.7 or newer, or pre-Version 9.7.
- z/OS There is currently no solution to enforce wait for outcome behavior when the Apply program on z/OS uses LOAD from CURSOR to perform the full refresh from a DB2 V9.7 source database. In this case, the best solution is to suspend any applications that update the source table from the time the registration is activated until the full refresh begins.

Procedure

To ensure that utilities used for full refresh wait for committed data, use one of the following procedures depending on the utility:

Utility used with ASNLOAD	**Procedure**

LOAD from CURSOR	**Apply is Version 9.7 or newer**
	Issue the following command at the Apply control server:
	`db2 alter server` *`server_name`*
	`OPTIONS(ADD CONCURRENT_ACCESS_RESOLUTION 'W');`
	This command is required even if DB2 is Version 9.7 or newer.
	
	Apply is pre-Version 9.7
	Note: If you are unable to follow this procedure, suspend any applications that update the source table during the beginning of the load.
	1. From the Apply control server, connect to the source database.
	2. Bind the SQL packages that are used for Call Level Interface (CLI) connections with a generic bind option into a specific package by using the following command:
	`db2 bind @db2cli.lst generic`
	`"CONCURRENTACCESSRESOLUTION WAIT_FOR_OUTCOME"`
	`COLLECTION ASN`
	3. Add the following name-value pair to the `db2cli.ini` file at the federated database, below the stanza that declares the options for the server definition to which the nickname belongs:
	`[`*`data_source_name`*`]`
	`CURRENTPACKAGESET=ASN`
	Where *data_source_name* is the source database that the `db2cli.bnd` packages were bound against.
	Recommendation: If you use a federated server for both replication and other purposes, create a new dedicated server for use by replication that has the CONCURRENT_ACCESS_RESOLUTION=W option set, and allow other applications to use the existing server name.

Utility used with ASNLOAD	**Procedure**
EXPORT/IMPORT EXPORT/LOAD	If you plan to use the DB2 EXPORT utility to refresh target tables from a DB2 source that is at Version 9.7 or newer, and the user ID that starts the Apply program does not have BINDADD authority, you must perform the following bind before Apply starts: `db2 bind @db2ubind.lst CONCURRENTACCESSRESOLUTION WAIT FOR OUTCOME COLLECTION ASN`

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

z/OS

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,
//PARM='V71A autostop LOGSTDOUT startmode=COLD
//capture_schema=JAY logreuse'
//STEPLIB DD DISP=SHR,DSN=DPROPR.ASN81 .SASNLOAD
//DD DISP=SHR,DSN=SYS1.SCEERUN
//DD DISP=SHR,DSN=DSN7.SDSNLOAD
//CEEDUMP DD SYSOUT=
//SYSPRINT DD SYSOUT=
//SYSTERM DD DUMMY
//
```

Using 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 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.	Prepare the JCL for z/OS by specifying the appropriate invocation parameters in the PARM field of the Replication Alert Monitor job. Customize the JCL to meet your site's requirements. A sample of invocation JCL in library SASNSAMP(ASNMON#) is included with the Replication Alert Monitor for z/OS.
	
	An example of this line in the invocation JCL is:
	`//monasn EXEC PGM=ASNMON,PARM='monitor_server=DSN`
	`                         monitor_qual=monqual'`
	
	where `DSN` is a subsystem name and `monqual` is the monitor qualifier.

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 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.

Working with running SQL replication programs by using the MVS MODIFY command

After you start the Capture program, Apply program, or Replication Alert Monitor program, you can use the **MODIFY** command to stop the program or to perform related tasks.

Procedure

To work with running programs on z/OS:

Run the **MODIFY** command from the z/OS console. You can use the abbreviation f, as shown in the following syntax example:

```
►►—f—jobname—,—┤ Parameters ├——————————————————————————————————►◄
```

f *jobname* , replaces the actual command name: **asnccmd**, **asnacmd**, or **asnmcmd**. The operational parameters that apply to each of the commands can be used with the f keyword.

For example, to stop an Apply program that uses the PLS job name, you use the following command:

```
F PLS,stop
```

Table 13 list the Capture commands that you can run with the f keyword. In all examples, the job name is mycap.

Table 13. Sample MODIFY commands for the Capture program

Parameter	Sample command that usesf keyword
prune	`f mycap,prune`
qryparms	`f mycap,qryparms`
reinit	`f mycap,reinit`
suspend	`f mycap,suspend`
resume	`f mycap,resume`
status	`f mycap,status`
stop	`f mycap,stop`

Table 13. Sample MODIFY commands for the Capture program (continued)

Parameter	Sample command that usesf keyword

chgparms	`f mycap,chgparms autostop=y`
	`f mycap,chgparms commit_interval=`*n*
	`f mycap,chgparms logreuse=y`
	`f mycap,chgparms logstdout=y`
	`f mycap,chgparms memory_limit=`*n*
	`f mycap,chgparms monitor_interval=`*n*
	`f mycap,chgparms monitor_limit=`*n*
	`f mycap,chgparms prune_interval=`*n*
	`f mycap,chgparms retention_limit=`*n*
	`f mycap,chgparms signal_limit=`*n*
	`f mycap,chgparms sleep_interval=`*n*
	`f mycap,chgparms term=y`
	`f mycap,chgparms trace_limit=`*n*

Table 14 list the Apply commands that you can run with the f keyword. In all examples, the job name is myapp.

Table 14. Sample MODIFY commands for the Apply program

Parameter	Sample command that uses f keyword
status	`f myapp,status`
stop	`f myapp,stop`

Table 15 lists asntrc program commands that you can run with the f keyword. In all examples, the job name is mycap.

Table 15. Sample MODIFY commands for the asntrc program

Task	Sample command that uses f keyword

Start a program trace with the **asntrc** command	`f mycap,asntrc on`
	`f mycap,asntrc statlong`

Format an asntrc fmt report and direct the output to a z/OS data set	`F mycap, asntrc fmt -ofn //'USRT001.TRCFMT'`
Format an asntrc flw report and direct the output to a z/OS data set	`F mycap, asntrc flw -ofn //'USRT001.TRCFLW'`
Stop a program trace	`F mycap, asntrc off`

Recommendation: Preallocate asntrc flw and fmt output files so that they are large enough to contain the asntrc reports. Use these attributes:

- **Data set name:** `USRT001.TRCFMT` or `USRT001.TRCFLW`
- **Primary allocated cylinders:** 2
- **Normal allocated extents:** 1
- **Data class:** None (Current utilization)
- **Used cylinders:** 2
- **Record format:** VB used extents: 1
- **Record length:** 1028
- **Block size:** 6144
- **1st extent cylinders:** 2
- **Secondary cylinders:** 1

- **SMS compressible:** NO

Table 16 list the Replication Alert Monitor commands that you can run with the f keyword. In all examples, the job name is mymon.

Table 16. Sample MODIFY commands for the Replication Alert Monitor program

Parameter	Sample command that uses f keyword
reinit	`f mymon,reinit`
status	`f mymon,status`
qryparms	`f mymon,qryparms`
suspend	`f mymon,suspend`
resume	`f mymon,resume`
stop	`f mymon,stop`
chgparms	`f mymon,chgparms monitor_interval=`*n* `f mymon,chgparms autoprune=y` `f mymon,chgparms trace_limit=`*n* `f mymon,chgparms alert_prune_limit=`*n* `f mymon,chgparms max_notifications_per_alert=`*n* `f mymon,chgparms max_notifications_minutes=`*n*

For information about **MODIFY**, see *z/OS MVS System Commands*.

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 Specifying your time zone.

   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 Automatic Restart Manager (ARM) to automatically restart replication and publishing (z/OS)

You can use the Automatic Restart Manager (ARM) recovery system on z/OS to restart the Q Capture, Q Apply, Capture, Apply, and Replication Alert Monitor programs.

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 program is APF authorized. For example, to use ARM with the Q Apply, Apply, or Replication Alert Monitor program, you must copy the appropriate load module into an APF authorized library. (The Q Capture and Capture programs must be APF authorized regardless of whether or not you are using ARM.)

About this task

ARM is a z/OS 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 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*.

Procedure

To use ARM to automatically restart replication and publishing programs:

1. Specify one of the following element names when you configure ARM:

Program	Element name
Q Capture	ASNQC*xxxxyyyy*
Q Apply	ASNQA*xxxxyyyy*
Capture	ASNTC *xxxxyyyy*
Apply	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.

2. Optional: If you have more than one instance of a replication or publishing program running within a data-sharing member, specify the **arm** parameter when you start the programs to create a unique ARM element name for each program instance.

   The **arm** parameter takes a three-character value that is appended to the element names that are listed in the previous table. The syntax is **arm**=*zzz*, where *zzz* can be any length of alphanumeric string. The replication program, however, will concatenate only up to three characters to the current name and pad with blanks, if necessary, to make a unique 16-byte name.

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 a z/OS subsystem. ARM restarts registered programs 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 Q Capture, Q Apply, Capture, Apply, or Replication Alert Monitor program with the parameter **term**=n, the program does not stop when DB2 is quiesced or stopped. In this case, the program does not de-register from ARM. It continues to run but does not perform its actual work until DB2 is unquiesced or started.

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	Desrcription

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.
	`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;`

Replication Center	Use one of the following windows: • 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**.
System i **`ADDDPRREG`** system command	Use the Add DPR registration (ADDDPRREG) command to register a new table on System i.

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:
	`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;`

Replication Center	Use one of the following windows: • 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 you use 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

- System i Altering a registered source table on System i to add a new column is not supported. System i creates an EJ (end journaling) journal entry before making the change to the source table on the ALTER operation. When you add a new column to a source table, you must drop and recreate the registration and subscription for the table. When you add columns to a System i table that uses a relative record number (RRN) as the primary key, remove the registration, add the column to the source table, and then add this table again as a new registration. Specify 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. Optional: If any of the columns that you added have default values, run the REORG utility on the source table.
11. z/OS Linux UNIX Windows If you are running the Apply program with **opt4one** set to y, stop and then restart the Apply program.
12. Reactivate the subscription sets.

Handling of ALTER TABLE ALTER COLUMN SET DATA TYPE operations

Starting with Version 10.1 on z/OS and Linux, UNIX, and Windows, SQL Replication handles ALTER TABLE ALTER COLUMN SET DATA TYPE operations at the source table by changing the data type of the corresponding CD table column. You must still alter the target table column.

Prerequisites:

- The Capture server must be at Version 10.1 or newer on both z/OS and Linux, UNIX, and Windows. The exception is extending a VARCHAR/VARGRAPHIC column. This operation is supported on older versions.
- z/OS Some configurations steps are required on z/OS. See Enabling replication of ADD COLUMN and SET DATA TYPE operations.

Capture also alters any before-image columns in the CD table to the new data type.

z/OS On z/OS, after the CD table is altered, DB2 puts the table into REORG PENDING state, which requires a REORG operation before the table can be used. The Capture program calls the DB2 stored procedure ADMIN_REVALIDATE_DB_OBJECTS to remove the table from REORG PENDING state. The exceptions to this REORG requirement are extending the length of VARCHAR or VARGRAPHIC columns.

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 by 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 by 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.

About this task

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

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

Procedure

To remove registrations by using the Replication Center:

1. Deactivate the subscription set to which the registered object belongs, or stop the Apply program. In the Replication Center, right-click the subscription set in the contents pane and click **Deactivate**.
2. Deactivate the registration that you want to delete to ensure that the Capture program finishes any current processing of this object, or stop the Capture program. To deactivate, right-click the registered object in the contents pane and click **Stop Capturing Changes**.
3. Remove the subscription-set member for the registered object. Right-click the subscription set in the navigation tree and click **Delete Members**. Then use the Delete Members from Subscription Sets notebook to remove the member and, if wanted, the target table.
4. Remove the registration. Right-click the registered object in the contents pane, and select **Delete**.
5. Reinitialize the Capture program so that it picks up the changes. Right-click the Capture control server in the navigation tree and click **Reinitialize Capture**.
6. Activate the subscription set. Right-click the set in the contents pane and click **Activate Indefinitely**.
7. Start the Capture program, Apply program, or both if you stopped them.

System i On System i, use the RMVDPRREG (remove DPR registration) 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 by using the **asnmcmd reinit** system command. You can also stop the monitors by using the **asnmcmd stop** system command and then restart the programs by 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 (System i 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 by using the Replication Center or ASNCLP.
8. Restart the Capture program with the new schema name.
9. Reactivate the associated subscription sets by 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**.
System i **ADDDPRSUB** system command	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 System i, 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**.
System i `ADDDPRSUBM` system command	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

ASNCLP command-line program	Use the ALTER SUBSCRIPTION SET command.
	
	The following commands set the environment and change the subscription set SET00 to lower the timing interval to 15 minutes:
	`SET SERVER CAPTURE TO DB SAMPLE;`
	`SET SERVER CONTROL TO DB TARGET;`
	`SET OUTPUT CAPTURE SCRIPT "capsubsetchg.sql"`
	`CONTROLSCRIPT "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;`

Replication Center	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**.

3. Reactivate the subscription set.

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

Changing subscription set names

You can change the name of a subscription set without having to drop and recreate 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 by using the **`asnmcmd reinit`** system command. You can also stop the monitors by using the **`asnmcmd stop`** command and then restart the programs by using the **`asnmon`** 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 with 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 by using the **`asnmcmd reinit`** system command. You can also stop the monitors by using the **`asnmcmd stop`** command and then restart the programs by 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,..Stmtn);
      ```

      where *NewName* is the name of the new subscription set that you are creating, *ApplyQual* is the Apply qualifier, *ExistName* is the name of the existing subscription set being split, *Val* is either F or S, and *Stmt1*, *Stmt2*, and *Stmtn* correspond to the numbers of the statements that you are moving to the new subscription set.
   b. Adjust the AUX_STMTS column values in the IBMSNAP_SUBS_SET table to reflect the new count of statements for both subscription sets. Renumber the statements to eliminate any duplicates, if necessary.
8. From the Capture control server, run the following SQL statement to change the name of the subscription set in the 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. z/OS Linux UNIX Windows If you are running the Apply program with **`opt4one`** set to `y`, 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 synch point 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 synch point and synchtime 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. z/OS Linux UNIX Windows If you are running the Apply program with **opt4one** set to `y`, stop and then restart the Apply program.

9. Reactivate the merged subscription set 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 recreating 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 that use 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 by using the **asnmcmd reinit** system command. You can also stop the monitors by using the **asnmcmd stop** command and then restart the programs by 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 by 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 by 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	Use the DROP SUBSCRIPTION SET command.
	
	The following commands set the environment and drop a subscription set named SET00 with an Apply qualifier of AQ00.
	`SET SERVER CAPTURE TO DB SAMPLE;`
	`SET SERVER CONTROL TO DB TARGET;`
	`SET OUTPUT CAPTURE SCRIPT "drpcapsubset.sql"`
	`CONTROLSCRIPT "drpappsubset.sql";`
	`SET LOG "drpsubset.err";`
	`SET RUN SCRIPT LATER;`
	`DROP SUBSCRIPTION SET SETNAME SET00 APPLYQUAL AQ00;`

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**.
System i `RMVDPRSUB` system command	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 by 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 with 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 recreate your source table, or reorganize and compress your source table without setting 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 you use 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 by using the following SQL statement:

   ```
   INSERT INTO Schema.IBMSNAP_SIGNAL
             (signal_type,
              signal_subtype,
              signal_state)
         VALUES('CMD',
               'STOP',
               'P');
   ```

   The Capture program acts on this signal table log record after the Capture program finds this record on the database recovery log and only when the Capture program finds the corresponding commit record for this insert, verifying that this event was committed.

   The Capture program shuts down all Capture threads in an orderly manner after committing all captured data from the transactions on the log that is prior to the commit log record for the DB2 unit of work that contains this inserted IBMSNAP_SIGNAL row. Before terminating, the Capture program also updates the following values in the IBMSNAP_SIGNAL table row that corresponds to the insert log record being processed:

   - SIGNAL_STATE = 'R' (received by the Capture program)
   - SIGNAL_LSN = the log sequence number from the commit log record for the DB2 unit of work that contains this signal row insert

   All log records for the source database are processed by the Capture program when it terminates.

2. Run the source database backup or image copy utilities.
3. Use the value in the SIGNAL_LSN column from the IBMSNAP_SIGNAL table row that you inserted as an END_SYNCHPOINT value in the IBMSNAP_SUBS_EVENT table. This value alerts the Apply program that all the data committed prior to the backup point has been collected by the Capture program and that the Apply program should fetch and apply data only up to the value of the SIGNAL_LSN column. The subscription sets process all data up to the SIGNAL_LSN value.
4. Run the target database backup or image copy utilities. The source and target databases now have equivalent recovery points, and you can recover both databases at a common point of consistency.

You can resume all source database activity as soon as the Apply events have been set and the source database backup or image copy utility activity is complete. You can also start the Capture program. After the target database backup or image copy utility activity is complete, you can change the scheduling options of your subscription sets back to their original settings (time-based, event-based, or both).

System i 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 the *schema*.IBMSNAP_SIGNAL table. 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 by 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 by 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: Optional: Remove the registration.
3. System i Optional: 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 (System i)

On System i, 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

System i 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 System i 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 following example demonstrates the use of this procedure:

   a. At 12:00 you stop the Capture program and all applications.
   b. You wait until 13:00 so that the journal entry timestamps only have values up to 12:00.
   c. You set the system time back to 12:00.
   d. You make a change. The journal entry timestamp for the change will be 12:01.
   e. You restart Capture. Capture will start from 12:00 and therefore will capture the change that came at 12:01 (Daylight Savings Time), which is 13.01 in Standard time.

   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 or later 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 all supported 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 System i remote journals.

Chapter 13. Maintaining an 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 System i), 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 the Apply program performs a full refresh.

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\SQLOGDIR\
   ```
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 either the SQLOGCTL.LFH.1 file or its mirror copy, SQLOGCTL.LFH.2. Both files are located in the database directory. The system retrieves and displays the name of the file that contains the log record that is identified by the log sequence number. For example:

```
Given LSN is contained in the log file S000123.LOG
```

Access to journal receivers (System i)

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 System i. 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 might 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 might 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:

- System i IBMSNAP_APPLY_JOB
- 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.

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 (z/OS, Linux, UNIX, Windows)

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 with 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
z/OS **REORG utility with the PREFORMAT option**	The PREFORMAT option of this utility speeds up the insert processing of the Capture program.
System i **RGZPFM (Reorganize Physical File Member) command**	You can reorganize the UOW table and active CD tables when the Capture program ends by specifying the **RGZCTLTBL(*YES)** parameter on the **`ENDDPRCAP`** command.
Linux UNIX Windows **`REORG` command**	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
- System i IBMSNAP_AUTHTKN

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 **`autoprune`** parameter to yes by using one of the following methods:

Method	Description
Start a Capture program with automatic pruning.	Issue the **`asncap`** system command with **`autoprune`**=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 **`asnccmd chgparms`** command with **`autoprune`**=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 **`asnccmd`** 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 synch point value over all Apply programs that subscribe to each CD table and on the minimum overall synch point 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.

z/OS Linux UNIX Windows When a Capture program starts with the warmns or warmsi option, 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.

- System i 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 by 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 (System i) determine how long rows remain in the IBMSNAP_CAPMON table
- The **`trace_limit`** parameter (Linux, UNIX, Windows, z/OS) and the **`TRCLMT`** parameter (System i) 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 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 the 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. Comparing and repairing tables

The **asntdiff** and **asntrep** commands detect and repair differences between tables. In Q Replication and SQL Replication, the commands enable you to find differences quickly and synchronize tables without performing a full refresh, or load, of the target table.

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** commands 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 compare and repair utilities do not depend on logs, triggers, or isolation level.

Procedure

To compare and repair tables, run the **asntdiff** command, and then run the **asntrep** command.

Table compare utility (asntdiff)

The **asntdiff** command compares the columns in one table to their corresponding columns in another table and generates a list of differences between the two in the form of a DB2 table.

To use the compare 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 also use SQL statements in an input file to specify the tables to compare.

The following sections explain how to use the asntdiff command:
- "Overview of the asntdiff command"
- "When to use the compare utility" on page 206
- "Where differences are stored" on page 206
- "Required authorizations" on page 207
- "Restrictions for key columns at source and target" on page 208
- "Data type considerations" on page 208
- "Effects of filtering" on page 209
- "Comparisons based on queries instead of subscriptions" on page 210
- "Comparing a subset of table rows" on page 210

Overview of the asntdiff command

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 System i 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.

z/OS The ASNTDIFF sample job in the SASNSAMP data set provides further information that is specific to the z/OS platform.

For Q Replication, the target must be a user copy table or a consistent-change-data (CCD) table that is condensed and complete. Stored procedure targets are not supported. For SQL Replication, the target must be a user table, point-in-time table, replica table, user-copy table, or consistent-change-data (CCD) table that is condensed and complete.

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.

When to use the compare utility

The best time to use the 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.

Where differences are stored

The **asntdiff** command creates a difference table in the source database or subsystem to store differences that it finds.

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, existing table space by using the DIFF_TABLESPACE parameter.

The difference table has two or more columns. One column is named DIFF, with a blank space at the end on Linux, UNIX, and Windows. 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.

Required authorizations

These database authorizations are required for the compare utility:

- Access privileges to the tables that are being compared, and to the replication control tables unless the -f (file) option is used

- Linux UNIX Windows Read privileges for the password file if the PWDFILE keyword is used
- WRITE privileges for the directory that is specified by the DIFF_PATH keyword
- To create the difference table, CREATETAB authority on the source database and USE privilege on the table space. In addition, one of the following privileges is needed:
  - IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name of the table does not exist
  - CREATEIN privilege on an existing schema if the table is created in this schema

  On z/OS, if the user ID that runs asntdiff does not have authority to create tables, you can use the SQLID keyword to specify an authorization ID that can be used to create the difference table.
- DROPIN privilege on the schema to drop the difference table unless DIFF_DROP=N is used
- SELECT, DELETE, and INSERT privileges on the difference table (at the source). The default schema name is ASN and the default table name is ASNTDIFF.

Restrictions for key columns at source and target

The asntdiff utility supports multiple-byte character sets when the database is defined with SYSTEM or IDENTITY. However, the columns that are used as keys for replication at the source and target tables must use single-byte characters for the utility to compare the tables.

In a Linux, UNIX, or Windows database that uses Unicode, the characters in key data cannot be greater than the base U.S. English ASCII subset (first 256 ASCII characters) or the asntdiff utility cannot compare the tables.

Data type considerations

You need to consider the data types of the tables that you are comparing when using asntdiff.

Different data types in sources and targets

The compare utility can build 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 and lengths 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.

Unsupported data types

The compare utility does not support comparisons between the following data types:

Nonkey columns

DECFLOAT, BLOB_FILE, CLOB_FILE, DBCLOB_FILE

Key columns

DECFLOAT, BLOB, CLOB, DBCLOB, VARGRAPHIC, GRAPHIC, LONG_VARGRAPHIC, BLOB_FILE, CLOB_FILE, DBCLOB_FILE, XML

Comparing the GRAPHIC data type

Columns with the GRAPHIC data type at the source and target might not match when you use the 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 utility compares the GRAPHIC data in a consistent manner.

TIMESTAMP WITH TIMEZONE restriction

The compare utility does not support comparisons that involved the TIMESTAMP WITH TIMEZONE data type that was introduced in DB2 for z/OS Version 10.

Effects of filtering

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 or suppressed deletes.

Row filtering

The compare utility uses information from the replication control tables to avoid showing intentional differences:

SQL Replication

The utility uses the PREDICATES column in the IBMSNAP_SUBS_MEMBR 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.

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 might 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.

The asntdiff -f (input file) option does not support SUPPRESS_DELETES because it bases the table comparison on a SQL SELECT statement rather than the Q subscription definition.

Comparisons based on queries instead of subscriptions

The **asntdiff -f** command option enables you to do differencing by using SQL SELECT statements that are read from an input file. This option provides greater flexibility to do differencing between two generic tables. The **asntdiff -f** option does not use replication definitions to determine which tables and rows to compare as the standard asntdiff command does.

The **asntdiff -f** option works for all tables on Linux, UNIX, Windows, and z/OS. For details on this option, see "asntdiff –f (input file) command option" on page 281.

In addition to the SELECT statements, the input file contains the source and target database information, the difference table information, and optional parameters that specify methods for processing the differences. You can use a password file that is created by the **asnpwd** command to specify a user ID and password for connecting to the source and target databases.

Note: To compare DB2 XML columns by using the **asntdiff -f** option, you need to serialize the XML column as a character large-object (CLOB) data type by using the XMLSERIALIZE scalar function. For example, this SELECT statement in the input file compares the XMLColumn column in the source table Table 1 to the same column in another database table (the TARGET_SELECT would use the same function):

```
SOURCE_SELECT="select ID, XMLSERIALIZE(XMLColumn AS CLOB) AS XMLColumn
from Table1 order by 1"
```

Comparing a subset of table rows

You can use the **asntdiff** RANGECOL parameter to compare only some of the rows in the two tables. This parameter specifies a range of rows from the source table that are bounded by two timestamps. You provide the name of a DATE, TIME, or TIMESTAMP column in the source table, and then use one of three different clauses for specifying the range. When you compare tables that are involved in peer-to-peer replication, you can use the IBM-generated IBMQREPVERTIME column for the source column in the range clause.

The RANGECOL parameter is not valid for the **asntdiff -f** (input file) option. You can use a SQL WHERE clause in the input file to achieve similar results.

Running the asntdiff utility in parallel mode (z/OS)

By using the PARALLEL=Y option with the asntdiff command, you can run the table compare utility in a parallel mode that provides optimal performance when comparing very large tables.

In parallel mode, the asntdiff utility uses as many as 21 threads to compare the data in two specified tables, significantly decreasing processing time while maintaining the accuracy of the comparison. With this mode:

- The utility internally partitions the two tables and compares these partitions in parallel.

- Row retrieval for each partition pair occurs in parallel.
- The differences that are found in each partition pair are then combined to obtain the overall result.

Using this method reduces processing time and reduces memory use because it avoids materializing large intermediate results. The parallel mode also minimizes network traffic because the checksum calculations are pushed down to each database.

To use asntdiff in parallel mode, it is recommended but optional that the two tables have date, time, or timestamp columns and a unique index or primary key. Both tables must be on DB2 for z/OS and must use the same code page and collation sequence.

The following sections provide more detail about the use of parallel mode.
- "Installation requirements"
- "Required authorizations"
- "Restrictions" on page 212
- "Usage tips" on page 212

Installation requirements

To use the asntdiff utility in parallel mode, you must install a stored procedure (ASNTDSP) at the systems that contain any table to be compared. The ASNTDSP sample job is included in the SASNSAMP dataset. For best results, use an application environment with NUMTCB = 8 - 15.

The following code defines ASNTDSP:

```
CREATE PROCEDURE ASN.TDIFF
 ( IN     SELECTSTMT          VARCHAR(32700),
   IN     GTTNAME             VARCHAR(128),
   IN     OPTIONS             VARCHAR(1331),
   OUT    RETURN_BLOCK_CRC    VARCHAR(21),
   OUT    RETURN_NUM_ROWS     INTEGER,
   OUT    RETURN_CODE         INTEGER,
   OUT    RETURN_MESSAGE      VARCHAR(30000))
 PARAMETER CCSID EBCDIC
 EXTERNAL NAME ASNTDSP
 LANGUAGE C
 PARAMETER STYLE GENERAL WITH NULLS
 COLLID ASNTDIFF
 WLM ENVIRONMENT !!WLMENV4!!
 MODIFIES SQL DATA
 ASUTIME NO LIMIT
 STAY RESIDENT YES
 PROGRAM TYPE MAIN
 SECURITY USER
 RUN OPTIONS 'TRAP(OFF),STACK(,,ANY,),POSIX(ON)'
 COMMIT ON RETURN NO;
```

Required authorizations

These database authorizations and privileges or higher are required to run asntdiff in parallel mode:
- SELECT on the tables that are being compared.
- These privileges for the difference table:

  - CREATEIN and DROPIN on an existing schema if the table is created in this schema
  - If the parameter DIFF_TABLESPACE is not specified, CREATETAB and CREATETS authority on the default database DSNDB04. Note that on DB2 for z/OS Version 10, if the IN clause is not specified with CREATE TABLE, CREATETAB privilege on database DSNDB04 is required.
  - If the parameter DIFF_TABLESPACE is explicitly specified, CREATETAB authority on the database that contains the DIFF_TABLESPACE and USE privilege on the table space that is specified by DIFF_TABLESPACE
- To create the created global temporary tables in the work file databases, CREATETMTAB privileges at any databases that are involved
- SELECT, DELETE, and INSERT privileges on the database where the difference table is created. The default schema name is ASN and the default table name is ASNTDIFF.
- SELECT privileges on the catalog tables SYSIBM.SYSDUMMY1, SYSIBM.SYSTABLES, SYSIBM.SYSKEYS, and SYSIBM.SYSINDEXES
- EXECUTE privileges on procedure ASN.TDIFF
- EXECUTE privileges on package ASNTDIFF.ASNTDSP
- EXECUTE privileges on plan ASNRD101

Make sure that the necessary DB2 authorizations are granted to the user ID that runs asntdiff before you start the utility. You can use the TARGET_SQLID and SOURCE_SQLID parameters to change the value of CURRENT SQLID to an authorization ID that has sufficient authorities.

Restrictions

- The two tables that are being compared must have the same code page and collation sequence. Otherwise, use the PARALLEL=N option (the default) to compare the tables.
- When used in parallel mode, the asntdiff utility should be run from z/OS as a batch job that uses JCL.
- The tables that are being compared must reside on z/OS.
- Only the -F PARM option is supported when asntdiff runs in parallel mode.
- The supported SELECT statements that you use with the SOURCE_SELECT and TARGET_SELECT parameters must use this strucure:

  ```
  SELECT xxx FROM yyy (WHERE zzz) ORDER BY aaa
  ```

  The WHERE clause is optional.
- Supported data types for nonkey columns are DATE, TIME, TIMESTAMP, VARCHAR, CHAR, LONG VARCHAR, FLOAT, REAL, DECIMAL, NUMERIC, BIGINT, INTEGER, SMALLINT, ROWID, VARBINARY, BINARY, VARGRAPH, GRAPHIC, LONGRAPH.
- Supported data types for key columns are DATE, TIME, TIMESTAMP, VARCHAR, CHAR, LONG VARCHAR, FLOAT, REAL, DECIMAL, NUMERIC, BIGINT, INTEGER, SMALLINT, ROWID, VARBINARY, BINARY.

For other restrictions, see "asntdiff –f (input file) command option" on page 281.

Usage tips

- Columns that are used in WHERE clauses and ORDER BY clauses should use an index. The columns that you specify in the ORDER BY clause must follow the same index column order and ascending/descending attributes. Use the RUNSTATS and REORG utilities to keep table access information current.

- In parallel mode, the asntdiff utility does support a mix of ascending and descending order in the ORDER BY clause. The mix should be same as in the index. However, the utility might not give you optimal performance when the index uses this mixture. Results will still be correct.
- For optimal performance:
  - Increase the system resource limit for application threads and set NUMTHREADS to 21.
  - Do not use column alias and expressions against the key columns in SOURCE_SELECT and TARGET_SELECT.

Table repair utility (asntrep)

The **asntrep** command repairs differences between source and target tables on all DB2 servers by deleting, inserting, and updating rows. The command runs on Linux, UNIX, or Windows operating systems.

The **asntrep** command uses the difference table that is generated by the **asntdiff** command to take the following actions:

- 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.

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.

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. The repair utility does not support the use of an input file as does the compare utility.

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.

How the compare utility handles DB2 SQL compatibility features

DB2 for Linux, UNIX, and Windows Version 9.7 introduced SQL compatibility enhancements such as variable-length timestamps, the VARCHAR2 data type with special character string processing, and DATE-TIMESTAMP compatibility. Some considerations are required to use the **asntdiff** command with these new features.

The following sections describe these considerations:
- “Comparing TIMESTAMP non-key columns with different precisions ”
- “Comparing TIMESTAMP key columns with different precision”
- “Considerations when using the DATE data type as TIMESTAMP(0)” on page 216
- “Behavior when using the rangecol parameter ” on page 215
- “Compatibility option for text based strings ” on page 216
- “asntdiff file option (asntdiff –f) ” on page 217

Comparing TIMESTAMP non-key columns with different precisions

When asntdiff compares two tables that have TIMESTAMP columns of different precision, it creates a truncated version of the longer TIMESTAMP column and then compares the two equal-length values.

In the following example, Table A and Table B have TIMESTAMP columns of different lengths:

Table A	Table B

`Col1 - TIMESTAMP(6)`
`2009-02-05-12.46.01.126412`
	`Col1 - TIMESTAMP(12)`
	`2009-02-05-12.46.01.126412000000`

In this situation, asntdiff compares 2009-02-05-12.46.01.126412 from Table A with the truncated value of 2009-02-05-12.46.01.126412 from Table B, and reports matching values.

In the next example, Table A has a longer TIMESTAMP column than Table B because the target value was truncated as a result of replication (typically this occurs when the target database is pre-Version 9.7 and only supports the default TIMESTAMP precision of six-digits):

Table A	Table B

`Col1 - TIMESTAMP(12)`
`2009-02-05-12.46.01.126412123456`
	`Col1 - TIMESTAMP(6)`
	`2009-02-05-12.46.01.126412`

Here, asntdiff compares a truncated version of the source value, 2009-02-05-12.46.01.126412, with the target value of 2009-02-05-12.46.01.126412 and reports a match. Whenever asntdiff truncates a TIMESTAMP column, the utility issues warning message ASN4034W.

Comparing TIMESTAMP key columns with different precision

When asntdiff compares key columns with different TIMESTAMP precision, the same basic concepts hold: A version of the longer timestamp column is truncated to the length of the shorter timestamp column for the purpose of comparing.

In the following example, Table A has a TIMESTAMP(6) key column and a character column, and Table B has a TIMESTAMP(12) key column and a character column.

Table A	Table B
KEYCol1 - TIMESTAMP(6) 2009-02-05-12.46.01.126412	KEYCol1 - TIMESTAMP(12) 2009-02-05-12.46.01.126412000000
Col2 CHAR(12) "test String"	Col2 CHAR(12) "String test"

The utility compares the Table A key value of 2009-02-05-12.46.01.126412 with a truncated version of the Table B key value, 2009 -02-05-12.46.01.126412, and reports a match. It then compares the nonkey column values "test String" and "String test" and reports a "U 2" (update needed) in the difference table to signify 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:

DIFF	Col1 TIMESTAMP(6)
U 2	2009-02-05-12.46.01.126412

The second column in the difference table always contains the key value. Because the difference table DDL is based on the source table, asntdiff uses the source TIMESTAMP(6) value. If the source table had the longer TIMESTAMP column, for example a TIMESTAMP(12), the utility would truncate the TIMESTAMP(12) to a TIMESTAMP(6) in order to compare the keys. However, it would use the source table's TIMESTAMP(12) definition to create the difference table. The key value that is written to the difference table is, however, the key value that has been used during comparison: TIMESTAMP(6). This value is then padded to a TIMESTAMP(12).

In this situation, when you use the asntrep utility to repair differences between the source and target tables, asntrep assumes that the target key-column value is a result of replication, and thus if DB2 pads with 0s, a matching key on the target side is found and can be updated.

Behavior when using the rangecol parameter

The asntdiff **rangecol** invocation parameter, which enables you to compare a subset of rows in two tables based on a specified timestamp column, also requires special attention when the timestamp column is variable length and also a key column.

Table A

KEYCol1 - TIMESTAMP(12)	Col2 CHAR(12)
2009-02-05-12.46.01.126412123456	"test String"
2009-02-05-12.46.02.126412123456	"test String"
2009-02-05-12.46.03.126412123456	"test String"

Table B

KEYCol1 - TIMESTAMP(6)	Col2 CHAR(12)
2009-02-05-12.46.01.126412	"String test"
2009-02-05-12.46.02.126412	"String test"

KEYCol1 - TIMESTAMP(6)	Col2 CHAR(12)
2009-02-05-12.46.03.126412	"String test"

Using Table A and Table B above as examples, consider the following rangecol portion of an asntdiff invocation in which the TIMESTAMP(6) is used to specify which rows to compare:

```
RANGECOL="'KEYCol1' FROM: '2009-02-05-12.46.01.126412'
TO: '2009-02-05-12.46.03.126412'"
```

The range clause is rewritten by asntdiff into a SQL SELECT statement with a BETWEEN clause:

```
WHERE ("KEYCol1" BETWEEN '2009-02-05-12.46.01.126412'
AND '2009-02-05-12.46.03.126412')
```

To include all rows in the above scenario, use the source key values in the range clause. As a general rule, always use the longer TIMESTAMP column value in the range clause. For example, the following statement considers all six rows on both target and source side:

```
RANGECOL="'KEYCol1' FROM: '2009-02-05-12.46.01.126412123456'
TO: '2009-02-05-12.46.03.126412123456'"
```

Note: The scenarios described are only valid when the target table content has exclusively been populated by the Q Apply or Apply program. Any manual interaction with the target table could result in unexpected asntdiff results. As always, a thorough analysis of the results in the differencing table is required before you use the **asntrep** command to repair differences.

Considerations when using the DATE data type as TIMESTAMP(0)

The asntdiff utility does not support comparison of DATE and TIMESTAMP(0) data types. If the DATE data type compatibility feature is not enabled for the database that features the table with the DATE column, asntdiff gives the following message and terminates abnormally: "ASN4003E The data type or the length ... are not compatible."

The following example shows two databases, the second of which is enabled to use TIMESTAMP(0) columns for dates:

Database 1, Table A	Database 2 (compatibility vector 0x40), Table B
DATE 02/05/2009	TIMESTAMP(0) 2009-02-05-12.46.01

To compare these two tables, you must use the asntdiff file option and manually cast either of the two data types to a compatible data type.

Compatibility option for text based strings

With the compatibility option for character data enabled, an insert of an empty string into a text-based column results in a null value.

Database 1, Table A KEYCol1 - TIMESTAMP(6)	Database 1, Table A Col2 VARCHAR(12)
2009-02-05-12.46.01.126412	""

Database 2 (compatibility vector 0x20), Table B KEYCol1 - TIMESTAMP(6)	Database 2 (compatibility vector 0x20), Table B Col2 VARCHAR(12)
2009-02-05-12.46.01.126412	NULL

By default asntdiff flags a difference in Col2 and reports an update needed in the difference table. If you do not want asntdiff to report this as a difference, you can use the asntdiff file option with the following SQL statement in the SOURCE_SELECT parameter:

```
SELECT Col2 CASE WHEN Col2 = \'\' THEN NULL ELSE Col2 END FROM Database1
```

In any case, the warning message ASN4035W is issued once to make you aware of this scenario.

asntdiff file option (asntdiff –f)

To override any of the default behaviors mentioned above, it is recommended to employ the asntdiff file option that was introduced in Version 9.7.

The option allows you to use any SQL expression, for example you could use a CAST statement to avoid the truncation when comparing different length timestamp columns.

The following example pads the TIMESTAMP(6) to a TIMESTAMP(12):

```
SOURCE_SELECT= "SELECT CAST(KEYCol1 AS TIMESTAMP(12) ) AS KEYCol1, Col2
FROM TABLE_A ORDER BY 1"
TARGET_SELECT= "SELECT KEYCol1, Col2 FROM TABLE_B ORDER BY 1"
```

Chapter 15. 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 by using the commands that are available on your operating system.

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 17 shows commands that are used to start the replication programs at 3:00 p.m. on Friday:

Table 17. 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 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 (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 18. 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 by 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 System i operating system

You can schedule when to start the replication programs on the System i 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 16. Replication services (Windows)

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

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 19 describes the inputs for the replication service name.

Table 19. Inputs for the replication service name

Input	Description
instance	The name of the DB2 instance.
alias	The database alias of the Q Capture server, Q Apply server, Capture control server, Apply control server, or Monitor control server.
program	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).
qualifier_or_schema	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 Apply program that has the schema ASN and is working with database DB1 under the instance called INST1:

```
DB2.INST1.DB1.QAPP.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 QAPPLY 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 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.
- After you install DB2, you must restart your Windows server before you start a replication service.

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 Q Apply control server and a Monitor control server, you can create ten 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 **`asnscrt`** 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 by 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

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 (**asnqacmd**, **asnqccmd**, **asnccmd**, **asnacmd**, or **asnmcmd**), 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, use the **asnlist** command. You can optionally use 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. 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 System i 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 System i, 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 by 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 18. Checking the status of the SQL Replication programs

The following topics describe methods you can use to check the status of your replication environment.

Checking the status of replication programs (z/OS, Linux, UNIX, Windows)

You can quickly assess the current status of the Capture program, Apply program, or Replication Alert Monitor program.

Use one of the following commands to check the status of the replication programs:

Capture program
: `asnccmd` system command, **status** parameter

Apply program
: `asnacmd` system command, **status** parameter

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 threads:
  - Worker thread
  - Administration thread
  - Pruning thread
  - Serialization thread
  - Transaction reader thread (if the `asynchlogrd` startup parameter is set to yes)
- The Apply program has the following threads:
  - Administration thread (depending on your maintenance level, this might not be a separate thread)
  - Worker thread
  - Serialization thread
  - Monitor thread (if the MONITOR_ENABLED column in the IBMSNAP_APPPARMS table is set to Y)
- 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

The Apply worker thread sets its state to "waiting for database" if it cannot connect to its Apply control server and if the Apply program was started with the **term**=n parameter. You can run the status command in **asnacmd** or MODIFY on z/OS to check whether the Apply worker thread is running but unable to connect to the control server.

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.

Checking the status of the Capture and Apply journal jobs (System i)

On DB2 for System i, 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 (**WRKSBMJOB**) system command or the Work with Job (**WRKJOB**) system command to locate the job. Find the job's joblog to verify that it completed successfully or to identify why it failed.

Monitoring the progress of the Capture program (System i)

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 19. Customizing and running SQL scripts for 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 by 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 20. Naming rules for SQL Replication objects

The following table lists the limits for names of replication objects.

Table 20. Name limits for replication objects

Object	Name limits
Source and target tables	Linux UNIX Windows Follow the naming rules for your database management system. System i Names cannot include blanks, asterisks (*), question marks (?), single quotation marks ('), double quotation marks ("), or a slash (/).
Source and target columns	Follow the naming rules for your database management system. (Note that all before-image columns have a one-character prefix added to them. To avoid ambiguous before-image column names, ensure that source column names are unique to 127 characters and that the before-image column names will not conflict with existing column names when the before-image character prefix is added to the column name.)
Subscription set	A subscription-set name can include any characters allowed by DB2 for varying-character (VARCHAR) columns. **Recommendation:** Follow the naming rules for DB2 table and column names. Because DB2 replication stores the subscription-set name in each replication control server, be sure that the name is compatible for all three servers' code pages.
Capture schema	The Capture schema can be a string of 128 or fewer characters.[1] z/OS **Subsystems that are running Version 8 compatibility mode or earlier:** 18 or fewer characters System i The Capture schema (CAPCTLLIB) can be a string of 10 or fewer alphanumeric characters[1].
Apply qualifier	z/OS Linux UNIX Windows The Apply qualifier can be a string of 18 or fewer characters[1]. System i The Apply qualifier can be a string of 18 or fewer characters but, because Apply jobs can be only up to 10 characters long, the first 10 characters must be unique for a given Apply qualifier[1].
Monitor qualifier	z/OS Linux UNIX Windows The Monitor qualifier can be a string of 18 or fewer characters[1].

Table 20. 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:
   - 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 (\).

Windows 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 21. 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 System i.

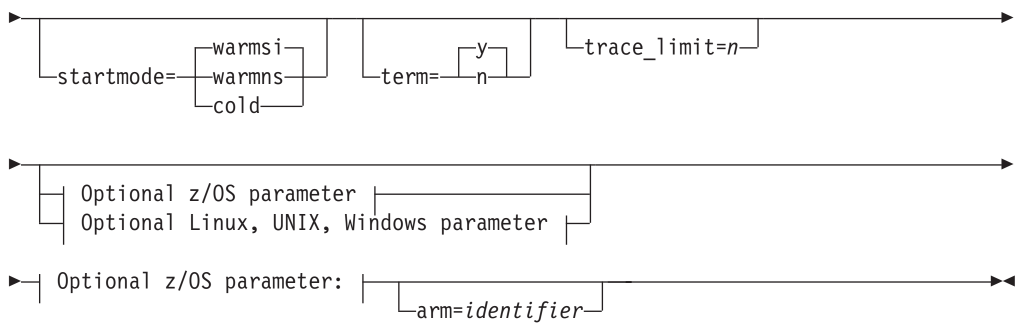
asncap: Starting Capture

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.

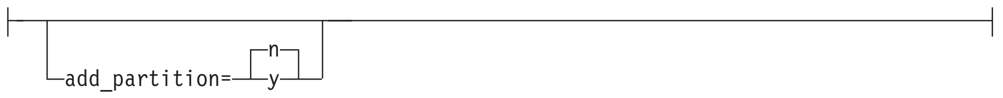
After you start the Capture program, it runs continuously until you stop it or it detects an unrecoverable error.

Syntax

```
>>──asncap──┬─────────────────────────┬──┬────────────────────────────┬──────────────>
            └─capture_server=db_name─┘  └─capture_schema=schema─┘

>──┬────────────────────┬──┬───────────────────────┬──┬───────────────────────┬──────>
   └─capture_path=path─┘  │               ┌─n─┐   │  │             ┌─y─┐     │
                          └─asynchlogrd=──┴─y─┴───┘  └─autoprune=──┴─n─┴─────┘

>──┬──────────────────────────────────┬──┬─────────────────────┬──┬──────────────┬───>
   │            ┌─n─┐                 │  └─commit_interval=n─┘  │      ┌─n─┐   │
   └─autostop=──┴─y─┴──┬───────────┬──┘                         └─hs=──┴─y─┴───┘
                       │      ┌─n─┐│
                       └─caf=─┴─y─┴┘

>──┬─────────────────────────────────┬──┬─────────────┬──┬────────────────────┬──────>
   └─ignore_transid=transaction_ID─┘  └─lag_limit=n─┘  │            ┌─n─┐   │
                                                        └─logreuse=──┴─y─┴───┘

>──┬──────────────────────┬──┬──────────────────┬──┬──────────────────────┬─────────>
   │             ┌─n─┐    │  └─memory_limit=n─┘  └─monitor_interval=n─┘
   └─logstdout=──┴─y─┴────┘

>──┬───────────────────┬──┬───────────────────┬──┬─────────────────────┬────────────>
   │           ┌─y─┐   │  └─monitor_limit=n─┘  └─part_hist_limit=n─┘
   └─migrate=──┴─n─┴───┘

>──┬──────────────────────────┬──┬────────────────────┬──┬────────────────────┬─────>
   │          ┌─asnpwd.aut─┐  │  └─prune_interval=n─┘  │            ┌─y─┐   │
   └─pwdfile=─┴─filename───┴──┘                         └─prunemsg=──┴─n─┴───┘

>──┬─────────────────────┬──┬────────────────────┬──┬─────────┬─────────────────────>
   └─retention_limit=n─┘  └─sleep_interval=n─┘  └─stale=n─┘
```



Optional Linux, UNIX, Windows parameter:



Parameters

Table 21 defines the invocation parameters.

Table 21. asncap invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems

Parameter	Definition
capture_server=*db_name*	Specifies the name of the Capture control server. 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. Linux UNIX Windows If you do not specify a Capture control server, this parameter defaults to the value from the DB2DBDFT environment variable.
add_partition=*y/n*	Linux UNIX Windows Specifies whether the Capture program starts reading the log file for the newly added partitions since the last time the Capture program was restarted. **n (default)** No new partitions have been added since the last time the Capture program was restarted. **y** The Capture program starts reading the log file on one or more of the new partitions. On each partition, the Capture program starts reading the log from the log sequence number (LSN) that was initially used the last time the database was started.

Table 21. asncap invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
arm=*identifier*	z/OS Specifies a three-character alphanumeric string that is used to identify a single instance of the Capture program to the Automatic Restart Manager. The value that you supply is appended to the ARM element name that Capture generates for itself: ASNTC*xxxxyyyy* (where *xxxx* is the data-sharing group attach name, and *yyyy* is the DB2 member name). You can specify any length of string for the **arm** parameter, but the Capture program will concatenate only up to three characters to the current name. If necessary, the Capture program will pad the name with blanks to make a unique 16-byte name.
asynchlogrd=*y/n*	**n (default)** Specifies that you want the Capture program to use the same thread for reading the DB2 recovery log and processing transactions that were captured from the log. **y** Specifies that you want the Capture program to use a dedicated thread for capturing transactions from the DB2 recovery log. The transaction reader thread prefetches committed transactions in a memory buffer, from which another thread gets the transactions and processes them into SQL statements for insertion into the CD table. This asynchronous mode can improve Capture throughput in all environments with particular benefits for partitioned databases and z/OS data-sharing. On systems with very high activity levels, this prefetching might lead to more memory usage. Adjust the **memory_limit** parameter accordingly. If you have a low volume of changes, you might prefer the default value of N to reduce CPU consumption.
capture_schema=*schema*	Specifies the name of the Capture schema that is used to identify a particular Capture program. The schema name that you enter must be 1 to 128 characters in length. The default is ASN.
capture_path=*path*	Specifies the location of the work files used by the Capture program. The default is the directory where the **asncap** command was invoked.
autoprune=*y/n*	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. **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. **n** Automatic pruning is disabled.

Table 21. asncap invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
autostop=*y/n*	Specifies whether the Capture program terminates after retrieving all the transactions that were logged before the Capture program started. **n (default)** The Capture program does not terminate after retrieving the transactions. **y** The Capture program terminates after retrieving the transactions.
caf=n/y	z/OS The Capture program runs with the default of Recoverable Resource Manager Services (RRS) connect (CAF=n). You can override this default and prompt the Capture program to use the Call Attach Facility (CAF) by specifying the **caf**=y option. The **caf**=y option specifies that the replication program overrides the default RRS connect and runs with CAF connect. **n (default)** The Capture program uses Recoverable Resource Manager Services (RRS) connect (CAF=n). **y** Specifies the replication program overrides the default RRS connect and runs with CAF connect. If RRS is not available you receive a message and the replication program switches to CAF. The message warns that the program was not able to connect because RRS is not started. The program attempts to use CAF instead. The program runs correctly with CAF connect.
commit_interval=*n*	Specifies the number of seconds that the Capture program waits before committing rows to the unit-of-work (UOW) and change-data (CD) tables. The default is 30 seconds.
ignore_transid=*transaction_ID*	Specifies that the Capture program will not capture the transaction that is identified by *transaction_ID*. The value for *transaction_ID* is a 10-byte hexadecimal identifier in the following format: z/OS 0000:*xxxx*:*xxxx*:*xxxx*:*mmmm* Where *xxxx*:*xxxx*:*xxxx* is the transaction ID, and *mmmm* is the data-sharing member ID. You can find the member ID in the last 2 bytes of the log record header in the LOGP output. The member ID is 0000 if data-sharing is not enabled. Linux UNIX Windows *nnnn*:0000:*xxxx*:*xxxx*:*xxxx* Where *xxxx*:*xxxx*:*xxxx* is the transaction ID, and *nnnn* is the partition identifier for partitioned databases (this value is 0000 if for non-partitioned databases).

Table 21. asncap invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
lag_limit=*n*	Specifies the number of minutes that the Capture program is allowed to lag in processing log records. The default is 10080 minutes (seven days). The Capture program checks the value of this parameter only during a warm start. If this limit is exceeded, the Capture program will not start.
logrdbufsz=*n*	Specifies the size of the buffer that the Capture program passes to DB2 when Capture retrieves log records. DB2 fills the buffer with available log records that Capture has not retrieved. The default value for DB2 for z/OS is 66KB; for DB2 for Linux, UNIX, and Windows the default is 256KB. For partitioned databases, Capture allocates a buffer of the size that is specified by **logrdbufsz** for each partition.
logread_prefetch=y/n	Linux UNIX Windows Specifies whether the Capture program uses separate threads to prefetch log records from each partition in a partitioned database. **n (default for nonpartitioned databases)** A single Capture log reader thread connects to all partitions. **y (default for partitioned databases)** A separate log reader thread connects to each partition. Using separate threads can increase Capture throughput but might increase CPU usage.
logreuse=*y/n*	Specifies whether the Capture program reuses or appends messages to the log file. **n (default)** The Capture program appends messages to the log file, even after the Capture program is restarted. **y** The Capture program reuses the log file by first truncating the current log file and then starting a new log when the Capture program is restarted. z/OS The log file name does not contain the DB2 instance name: `*capture_server*.*capture_schema*.CAP.log`. Linux UNIX Windows The log file name includes the DB2 instance name: `*db2instance*.*capture_server*.*capture_schema*.CAP.log.`
logstdout=*y/n*	Specifies where the Capture program directs the log file messages: **n (default)** The Capture program directs most log file messages to the log file only. Initialization messages go to both the log file and the standard output (STDOUT). **y** The Capture program directs log file messages to both the log file and the standard output (STDOUT).

Table 21. asncap invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
memory_limit=*n*	Specifies the maximum size (in megabytes) of memory that the Capture program can use to build transactions. After reaching this memory limit, the Capture program spills transactions to a file. The default is 32 megabytes. z/OS If you specify **memory_limit**=0, the Capture program determines the amount of memory to use from the region size parameter of the Capture job. The memory allocation is 80 percent of the region size.
migrate=y/n	Linux UNIX Windows Specifies that the Capture program starts from the beginning of the log after DB2 is migrated. **Important:** Use this option only the first time that Capture is started and specify **startmode**=warmns. Also, do not use the **migrate** parameter when you are migrating to Version 10.1 or later.
monitor_interval=*n*	Specifies how frequently (in seconds) the Capture program inserts rows into the IBMSNAP_CAPMON table. The default is 300 seconds (five minutes).
monitor_limit=*n*	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).
part_hist_limit=*n*	Specifies how long you want old data to remain in the IBMQREP_PART_HIST table before it becomes eligible for pruning. The default is 10080 minutes (seven days). This parameter also controls how far back in the log you can restart the Capture program because Capture uses IBMQREP_PART_HIST to determine what log records to read for a partitioned source table.
pwdfile=*filename*	Specifies the name of the password file. If you do not specify a password file, the default is asnpwd.aut. This command searches for the password file in the directory specified by the **capture_path** parameter. If no **capture_path** parameter is specified, this command searches for the password file in the directory where the command was invoked.
prune_interval=*n*	Specifies how frequently (in seconds) the change-data (CD), unit-of-work (UOW), 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).
prunemsg=y/n	Specifies whether the Capture program issues informational messages about the status of pruning. **y (default)** Capture issues informational messages about pruning status. **n** Capture does not issue informational messages about pruning status.

Table 21. asncap invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
retention_limit=*n*	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).
sleep_interval=*n*	Specifies the number of seconds that the Capture program sleeps when it finishes processing the active log and determines that the buffer is empty. The default is five seconds. z/OS Specifies the number of seconds that the Capture program sleeps after the buffer returns less than half full.
stale=*n*	Specifies the number of seconds that the Capture program waits to issue a warning message or take other action after it detects a long-running transaction with no commit or rollback log record. The program behavior depends on the platform of the source. On z/OS, Capture issues warning messages if has not seen a commit or rollback record for one hour (**stale**=3600). On both z/OS and Linux, UNIX, and Windows, if a transaction has been running for the number of seconds that are specified by **stale** and Capture did not see any row operations in the log for the transaction, it issues warning messages, does not replicate the transaction, and advances the log sequence number that it considers to be the oldest "in-flight" transaction that was not committed or rolled back. If some rows were captured for the transaction, only warning messages are issued.

Table 21. asncap invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
`startmode`=*mode*	Specifies the processing procedure used by the Capture program during a Capture startup. **warmsi (default)** The Capture program resumes processing where it ended in its previous run if warm start information is available. If this is the first time that you are starting the Capture program, it automatically switches to a cold start. During a warm start, the Capture program leaves the 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. **warmns** The Capture program resumes processing where it ended in its previous run if warm start information is available. If errors occur after the Capture program started, the Capture program terminates. If the Capture program cannot warm start, it does not switch to a cold start. **cold** The Capture program starts by deleting all rows in its CD and UOW tables. Most registrations are reset so that all subscriptions to those sources are fully refreshed during the next Apply processing cycle. Registrations for external CCDs and those subscriptions whose targets are noncomplete CCDs are not fully refreshed.
`term`=*y/n*	Specifies whether the Capture program terminates if DB2 is quiesced or stopped. **y (default)** The Capture program terminates if DB2 is quiesced or stopped. **n** The Capture program continues running if DB2 is quiesced or stopped. When DB2 initializes, the Capture program starts capturing at the point where it left off when DB2 was quiesced or stopped. If DB2 terminates through FORCE or abnormally terminates, the Capture program terminates even if you set this parameter to `n`. If you set this parameter to `n` and start DB2 with restricted access (ACCESS MAINT), the Capture program cannot connect and subsequently terminates.
`trace_limit`=*n*	Specifies how long (in minutes) a row can remain in the IBMSNAP_CAPTRACE table before it becomes eligible for pruning. All IBMSNAP_CAPTRACE rows that are older than the value of the **`trace_limit`** parameter are pruned at the next pruning cycle. The default is 10 080 minutes (seven days).

Return codes

The **asncap** command returns a zero return code upon successful completion. A nonzero return code is returned if the command is unsuccessful.

Examples for asncap

The following examples illustrate how to use the **asncap** command.

Example 1

To start a Capture program for the first time that uses a Capture control server named db and a Capture schema of ASN with work files located in the /home/files/capture/logs/ directory:

```
asncap capture_server=db capture_schema=ASN
  capture_path=/home/files/capture/logs/ startmode=cold
```

Example 2

To restart a Capture program without pruning after the Capture program was stopped:

```
asncap capture_server=db autoprune=n sleep_interval=10 startmode=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
   ```

asnccmd: Operating Capture

Use the **asnccmd** 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

For information on using the MVS MODIFY command to send commands to a running Capture program on z/OS, see Working with running SQL Replication programs by using the MVS MODIFY command.

```
►►──asnccmd──┬──────────────────────────┬──┬───────────────────────────┬──────────────►
             └─capture_server=db_name─┘  └─capture_schema=schema─┘

►──┬─chgparms──┤ parameters ├─┬──────────────────────────────────────────────────────►◄
   ├─prune────────────────────┤
   ├─qryparms─────────────────┤
   ├─reinit───────────────────┤
   ├─suspend──────────────────┤
   ├─resume───────────────────┤
   ├─status───────────────────┤
   └─stop─────────────────────┘
```

Parameters:

```
├──┬───────────────────────┬──┬──────────────────────┬──┬──────────────────────┬──────►
   │             ┌─y─┐     │  │            ┌─n─┐     │  └─commit_interval=n─┘
   └─autoprune=──┴─n─┴─────┘  └─autostop=───┴─y─┴─────┘

►──┬──────────────────────┬──┬───────────────────────┬──┬───────────────────┬─────────►
   │            ┌─n─┐     │  │             ┌─n─┐     │  └─memory_limit=n─┘
   └─logreuse=──┴─y─┴─────┘  └─logstdout=───┴─y─┴─────┘

►──┬──────────────────────┬──┬──────────────────┬──┬──────────────────────┬──────────►
   └─monitor_interval=n─┘  └─monitor_limit=n─┘  └─prune_interval=n─┘

►──┬─────────────────────┬──┬─────────────────────┬──┬─────────────────┬─────────────►
   └─retention_limit=n─┘  └─sleep_interval=n─┘  │        ┌─y─┐     │
                                                  └─term=──┴─n─┴─────┘

►──┬─────────────────┬──────────────────────────────────────────────────────────────┤
   └─trace_limit=n─┘
```

Parameters

Table 22 defines the invocation parameters for the asnccmd command.

Table 22. Definitions for asnccmd invocation parameters

Parameter	Definition
capture_server=*y/n*	Specifies the name of the Capture control server. 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. Linux UNIX Windows If you do not specify a Capture control server, this parameter defaults to the value from the *DB2DBDFT* environment variable.
capture_schema=*schema*	Specifies the name of the Capture schema that is used to identify a particular Capture program. The schema name must be 1 to 128 characters in length. The default is ASN.
chgparms	Specify to change one or more of the following operational parameters of a Capture program while it is running: • **autostop** • **commit_interval** • **logreuse** • **logstdout** • **memory_limit** • **monitor_interval** • **monitor_limit** • **prune_interval** • **retention_limit** • **signal_limit** • **sleep_interval** • **term** • **trace_limit** **Restriction:** z/OS The value of **memory_limit** cannot be altered with the Capture program is running. To change the value you must first stop the Capture program. You can specify multiple parameters in one **asnccmd chgparms** command, and you can change these parameter values as often as you want. The changes temporarily override the values in the IBMSNAP_CAPPARMS table, but they are not written to the table. When you stop and restart the Capture program, it uses the values in IBMSNAP_CAPPARMS. "asncap: Starting Capture" on page 239 includes descriptions of the parameters that you can override with this command.
prune	Specify this parameter if you want to prune the change-data (CD), unit-of-work (UOW), IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_SIGNAL tables once. The Capture program issues a message when the command is successfully queued.

Table 22. Definitions for asnccmd invocation parameters (continued)

Parameter	Definition
`qryparms`	Specify if you want the current operational parameter values written to the standard output (stdout).
`reinit`	Specify to have the Capture program obtain newly added replication sources from the IBMSNAP_REGISTER table. For example, use this parameter if you add a new replication source or if you use the ALTER ADD statement to add a column to a replication source and change-data (CD) table while the Capture program is running.
`suspend`	Specify to relinquish operating system resources to operational transactions during peak periods without destroying the Capture program environment. **Attention:** Do not suspend Capture to cancel a replication source. Instead, stop the Capture program.
`resume`	Specify to have a suspended Capture program resume capturing data.
`status`	Specify to receive messages that indicate the state of each Capture thread (administration, pruning, serialization, and worker).
`stop`	Specify to stop the Capture program in an orderly way and commit the log records processed up to that point.

Examples for asnccmd

The following examples illustrate how to use the **`asnccmd`** command.

Example 1

To enable a running Capture program to recognize newly added replication sources:

```
asnccmd capture_server=db capture_schema=ASN reinit
```

Example 2

To prune the CD, UOW, IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_SIGNAL tables once:

```
asnccmd capture_server=db capture_schema=ASN prune
```

Example 3

To receive messages about the state of each Capture thread:

```
asnccmd capture_server=db capture_schema=ASN status
```

Example 4

To send the current operational values of a Capture program to the standard output:

```
asnccmd capture_server=db capture_schema=ASN qryparms
```

Example 5

To disable the automatic pruning in a running Capture program:

```
asnccmd capture_server=db capture_schema=ASN chgparms autoprune=n
```

Example 6

To stop a running Capture program:

```
asnccmd capture_server=db capture_schema=ASN stop
```

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 or it detects an unrecoverable error.

Syntax

```
►►─asnapply──┬─┤ Required z/OS parameters ├──────────────────┬──────────────►
             └─┤ Required Linux, UNIX and Windows parameter ├─┘

►─┬───────────────────────┬─┬──────────────────────┬────────────────────────►
  └─control_server=db_name─┘ └─apply_path=pathname─┘

►─┬──────────────────────────────┬─┬────────────────┬─┬─────────────────┬───►
  │          ┌─asnpwd.aut─┐      │ │          ┌─n─┐ │ │           ┌─n─┐ │
  └─pwdfile=─┴─filename───┴──────┘ └─logreuse=─┴─y─┴─┘ └─logstdout=─┴─y─┴─┘

►─┬────────────────┬─┬───────────────┬─┬───────────────┬────────────────────►
  │         ┌─n─┐  │ │        ┌─y─┐  │ │        ┌─n─┐  │
  └─loadxit=─┴─y─┴──┘ └─inamsg=─┴─n─┴──┘ └─notify=─┴─y─┴──┘

►─┬──────────────────┬─┬──────────────┬─┬──────────────────┬────────────────►
  │          ┌─n─┐   │ │       ┌─y─┐  │ │          ┌─n─┐   │
  └─copyonce=─┴─y─┴───┘ └─sleep=─┴─n─┴──┘ └─trlreuse=─┴─y─┴───┘

►─┬─────────────────┬─┬─────────┬─┬───────────┬─────────────────────────────►
  │                 │ └─delay=n─┘ └─errwait=n─┘
  │         ┌─n─┐   │
  └─opt4one=─┴─y─┴───┘

►─┬───────────────────────────────────────────┬─────────────────────────────►
  │      ┌─y─┐                                │
  └─term=─┴─n─┴─┬────────────────────────────┬─┘
                └─refresh_commit_cnt=n──────┘

►─┬─────────────────────────────────────────────┬──────────────────────────►◄
  ├─┤ Optional z/OS parameters ├────────────────┤
  └─┤ Optional Linux, UNIX and Windows parameters ├─┘
```

Required z/OS parameters:

```
├──apply_qual=apply_qualifier──db2_subsystem=name──────────────────────────┤
```

Required Linux, UNIX and Windows parameter:

```
├──apply_qual=apply_qualifier──────────────────────────────────────────────┤
```

Optional z/OS parameters:

```
├──┬────────────────────────────┬──┬──────────────────┬──┤
   │                ┌─mem──┐    │  └─arm=identifier───┘
   └─spillfile=─────┴─disk─┴────┘
```

Optional Linux, UNIX and Windows parameters:

```
├──┬───────────────────────────────────────────┬──┬─────────────────────┬──┤
   │                   ┌─n─┐                    │  │           ┌─disk─┐  │
   └─sqlerrcontinue=───┴─y─┴──┬──────────────┬──┘  └─spillfile=─┴──────┴──┘
                              │      ┌─y─┐   │
                              └─caf=─┴─n─┴───┘
```

Parameters

Table 23 defines the invocation parameters.

Table 23. asnapply invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems

Parameter	Definition
apply_qual=*apply_qualifier*	Specifies the Apply qualifier that the Apply program uses to identify the subscriptions sets to be served. 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.
db2_subsystem=*name*	z/OS Specifies the name of the DB2 subsystem where the Apply program will run. The subsystem name that you enter can be a maximum of four characters. There is no default for this parameter. This parameter is required.
control_server=*db_name*	Specifies the name of the Apply control server on which the subscription definitions and Apply program control tables reside. z/OS Specifies the location name of the Apply control server. Linux UNIX Windows If you do not specify an Apply control server, this parameter defaults to the value from the DB2DBDFT environment variable.
apply_path=*pathname*	Specifies the location of the work files used by the Apply program. The default is the directory where the **asnapply** command was invoked.
pwdfile=*filename*	Specifies the name of the password file. If you do not specify a password file, the default is asnpwd.aut. This command searches for the password file in the directory specified by the **apply_path** parameter. If no **apply_path** parameter is specified, this command searches for the password file in the directory where the command was invoked.

Table 23. asnapply invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
`logreuse`=*y/n*	Specifies whether the Apply program reuses or appends messages to the log file. **n (default)** The Apply program appends messages to the log file, even after the Apply program is restarted. **y** The Apply program reuses the log file by deleting it and then re-creating it when the Apply program is restarted. z/OS The log file name does not contain the DB2 instance name: *`control_server`*`.`*`apply_qualifier`*`.APP.log`. Linux UNIX Windows The log file name contains the DB2 instance name: *`db2instance`*`.`*`control_server`*`.`*`apply_qualifier`*`.APP.log`.
`logstdout`=*y/n*	Specifies where the Apply program directs the log file messages: **n (default)** The Apply program directs most log file messages to the log file only. Initialization messages go to both the log file and the standard output (STDOUT). **y** The Apply program directs log file messages to both the log file and the standard output (STDOUT).
`loadxit`=*y/n*	Specifies whether the Apply program invokes ASNLOAD. ASNLOAD is an IBM-supplied exit routine that uses the export and load utilities to refresh target tables. **n (default)** The Apply program does not invoke ASNLOAD. **y** The Apply program invokes ASNLOAD.
`inamsg`=*y/n*	Specifies whether the Apply program issues a message when the Apply program is inactive. **y (default)** The Apply program issues a message when inactive. **n** The Apply program does not issue a message when inactive.
`notify`=*y/n*	Specifies whether the Apply program should invoke ASNDONE. ASNDONE is an exit routine that returns control to you when the Apply program finishes copying a subscription set. **n (default)** The Apply program does not invoke ASNDONE. **y** The Apply program invokes ASNDONE.

Table 23. asnapply invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
`copyonce`=*y/n*	Specifies whether the Apply program executes one copy cycle for each subscription set that is eligible at the time the Apply program is invoked. Then the Apply program terminates. An eligible subscription set meets the following criteria: • (ACTIVATE > 0) in the IBMSNAP_SUBS_SET table. When the ACTIVATE column value is greater than zero, the subscription set is active indefinitely or is used for a one-time-only subscription processing. • (REFRESH_TYPE = R or B) or (REFRESH_TYPE = E and the specified event occurred). The REFRESH_TYPE column value is stored in the IBMSNAP_SUBS_SET table. The MAX_SYNCH_MINUTES limit from the subscription sets table and the END_OF_PERIOD timestamp from the IBMSNAP_SUBS_EVENT table are honored if specified. **n (default)** The Apply program does not execute one copy cycle for each eligible subscription set. **y** The Apply program executes one copy cycle for each eligible subscription set.
`sleep`=*y/n*	Specifies how the Apply program is to proceed if no new subscription sets are eligible for processing. **y (default)** The Apply program sleeps. **n** The Apply program stops.
`trlreuse`=*y/n*	Specifies whether the Apply program empties the IBMSNAP_APPLYTRAIL table when the Apply program starts. **n (default)** The Apply program appends entries to the IBMSNAP_APPLYTRAIL table. The Apply program does not empty the table. **y** The Apply program empties the IBMSNAP_APPLYTRAIL table during program startup.
`opt4one`=*y/n*	Specifies whether the performance of the Apply program is optimized if only one subscription set is defined for the Apply program. **n (default)** The performance of the Apply program is not optimized for one subscription set. **y** The performance of the Apply program is optimized for one subscription set. If you set optimization to `y`, the Apply program caches and reuses the information about the subscription-set members. This reuse of subscription-set member information reduces CPU usage and improves throughput rates.

Table 23. asnapply invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
delay=*n*	Specifies the delay time (in seconds) at the end of each Apply cycle when continuous replication is used, where n=0, 1, 2, 3, 4, 5, or 6. The default is 6, and is used during continuous replication (that is, when the subscription set uses sleep=0 minutes). This parameter is ignored if **copyonce** is specified.
errwait=*n*	Specifies the number of seconds (1 to 65535) that the Apply program waits before retrying after the program encounters an error condition. The default value is 300 seconds (five minutes). **Note:** Do not specify too small a number, because the Apply program runs almost continuously and generates many rows in the IBMSNAP_APPLYTRAIL table.
term=*y/n*	Specifies whether the Apply program continues to run if it cannot connect to its control server. **y (default)** By default, the Apply program terminates if it cannot connect to its control server. **n** The Apply program does not terminate. Instead, Apply logs an error, waits for the amount of time set by the **errwait** parameter, then retries the connection. This parameter is ignored if **copyonce** is specified.
spillfile=*filetype*	Specifies where the fetched answer set is stored. z/OS Valid values are: **mem (default)** A memory file. If there is insufficient memory for the answer set, the Apply program uses a disk file. **disk** A disk file. **hs** High-performance data space (hiperspace) Linux UNIX Windows Valid values are: **disk (default)** A disk file.
arm=*identifier*	z/OS Specifies a three-character alphanumeric string that is used to identify a single instance of the Apply program to the Automatic Restart Manager. The value that you supply is appended to the ARM element name that Apply generates for itself: ASNTA*xxxxyyyy* (where *xxxx* is the data-sharing group attach name, and *yyyy* is the DB2 member name). You can specify any length of string for the **arm** parameter, but the Apply program will concatenate only up to three characters to the current name. If necessary, the Apply program will pad the name with blanks to make a unique 16-byte name.

Table 23. asnapply invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
`caf``=y/n`	z/OS Specifies whether the Apply program runs with Recoverable Resource Manager Services (RRS) connect (CAF=n). The runtime parameter Call Attach Facility (CAF) **`caf`** =y option specifies whether the replication program overrides RRS connect and runs with CAF connect. The **`caf`** =y option is the default for the Apply program. **y (default)** Specifies that the Apply program runs with CAF connect. **n** The Apply program uses Recoverable Resource Manager Services (RRS) connect (**`caf`** =n).
`sqlerrcontinue`=*y/n*	Specifies whether the Apply program continues processing when it encounters certain SQL errors. The Apply program checks the failing SQLSTATE against the values specified in the SQLSTATE file, which you create before running the Apply program. If a match is found, the Apply program writes the information about the failing row to an error file (`apply_qualifier.ERR`) and continues processing. The SQLSTATE file can contain up to 20 five-byte values. **n (default)** The Apply program does not check the SQLSTATE file. **y** The Apply program checks the SQLSTATE file during processing.
`refresh_commit_cnt`=*n*	During full refresh, Apply issues a COMMIT statement after the specified number of rows are inserted into the target table. Values can range from 0 to 134217727. A default value of 0 means that only one commit is issued after all rows have been inserted; no intermediate commits are issued.

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 with 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.

For information on using the MVS MODIFY command to send commands to a running Apply program on z/OS, see Working with running SQL Replication programs by using the MVS MODIFY command.

Syntax

```
►►──asnacmd──apply_qual=apply_qualifier──┬────────────────────────┬──────────►
                                         └─control_server=db_name─┘

►──┬─status─┬─────────────────────────────────────────────────────────────►◄
   └─stop───┘
```

Parameters

Table 24 defines the invocation parameters.

Table 24. asnacmd invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems

Parameter	Definition
apply_qual=*apply_qualifier*	Specifies the Apply qualifier that the Apply program uses to identify the subscriptions sets to be served. You must specify an Apply qualifier. The value that you enter must match the value of the APPLY_QUAL column in the IBMSNAP_SUBS_SET table. The Apply qualifier name is case sensitive and can be a maximum of 18 characters.

Table 24. asnacmd invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
control_server=*db_name*	Specifies the name of the Apply control server on which the subscription definitions and Apply control tables reside. z/OS The control server parameter is the name of the database server that connects to the control server. Linux UNIX Windows If you do not specify an Apply control server, this parameter defaults to the value from the DB2DBDFT environment variable.
status	Specify to receive messages that indicate the state of each thread (administration and worker) in Apply.
stop	Specify to stop the Apply program in an orderly way.

Examples for asnacmd

The following examples illustrate how to use the **asnacmd** command.

Example 1

To receive messages about the state of each Apply thread:

```
asnacmd apply_qual=AQ1 control_server=dbx status
```

Example 2

To stop the Apply program:

```
asnacmd apply_qual=AQ1 control_server=dbx stop
```

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 System i; 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

```
                         ┌──────────┐
                         ▼          │
►►──asnanalyze──-db───────db_alias──┴──┬──────────────────────────┬──┬─────────┬──────────►
                                       │          ┌─standard─┐   │  └─-tl──n──┘
                                       └─-la──────┼─detailed─┼───┘
                                                  └─simple───┘
```

```
►──┬──────┬──┬──────┬──┬──────┬──┬──────┬──┬───────────────────────────┬──►
   └─-at─n─┘  └─-ct─n─┘  └─-cm─n─┘  └─-sg─n─┘  │     ┌──────────────────┐  │
                                              └─-aq─▼─apply_qualifier─┴─┘

►──┬───────────────────┬──┬──────────────────────┬──┬─────────────────────┬──►
   └─-cs─capture_schema─┘  └─-od─output_directory─┘  └─-fn─output_filename─┘

►──┬────────────────────┬──────────────────────────────────────────────────►◄
   └─-pw─password_filepath─┘
```

Parameters

Table 25 defines the invocation parameters.

Table 25. asnanalyze invocation parameter definitions for Linux, UNIX and Windows operating systems

Parameter	Definition
-db *db_alias*	Specifies the Capture control server, target server, and Apply control server. You must provide at least one database alias. If there is more than one database alias, use blank spaces to separate the values.
-la *level_of_analysis*	Specifies the level of analysis to be reported: **standard (default)** Generates a report that includes the contents of the control tables and status information from the Capture and Apply programs. **detailed** Generates the information in the standard report and: • Change-data (CD) and unit-of-work (UOW) table pruning information • DB2 for z/OS table space partitioning and compression information • Analysis of target indexes for subscription keys **simple** Generates the information in the standard report, but excludes the detailed information from the IBMSNAP_SUBS_COLS table.
-tl *n*	Specifies the date range (0 to 30 days) of entries to be retrieved from the IBMSNAP_APPLYTRAIL table. The default is 3 days.
-at *n*	Specifies the date range (0 to 30 days) of entries to be retrieved from the Apply trace IBMSNAP_APPLYTRACE table. The default is 3 days.

Table 25. asnanalyze invocation parameter definitions for Linux, UNIX and Windows operating systems (continued)

Parameter	Definition
-ct *n*	Specifies the date range (0 to 30 days) of entries to be retrieved from the IBMSNAP_CAPTRACE table. The default is 3 days.
-cm *n*	Specifies the date range (0 to 30 days) of entries to be retrieved from the IBMSNAP_CAPMON table. The default is 3 days.
-sg *n*	Specifies the date range (0 to 30 days) of entries to be retrieved from the IBMSNAP_SIGNAL table. The default is 3 days.
-aq *apply_qualifier*	Specifies the Apply qualifier that identifies the specific subscription sets to be analyzed. You can specify more than one Apply qualifier. If there is more than one Apply qualifier, use blank spaces to separate the values. If no Apply qualifier is specified, all subscription sets for the specified database aliases are analyzed.
-cs *capture_schema*	Specifies the name of the Capture schema that you want to analyze. If you use this parameter, you can specify only one Capture schema.
-od *output_directory*	Specifies the directory in which you want to store the Analyzer report. The default is the current directory.
-fn *output_filename*	Specifies the name of the file that will contain the Analyzer report output. Use the file naming conventions of the operating system that you are using to run the Analyzer. If the file name already exists, the file is overwritten. The default file name is `asnanalyze.htm`.
-pw *password_filepath*	Specifies the name and path of the password file. If you do not specify this parameter, the Analyzer checks the current directory for the `asnpwd.aut` file.

Examples for asnanalyze

The following examples illustrate how to use the **asnanalyze** command.

Example 1

To analyze the replication control tables on a database called proddb1:

```
asnanalyze -db proddb1
```

Example 2

To obtain a detailed level of analysis about the replication control tables on the proddb1 and proddb2 databases:

```
asnanalyze -db proddb1 proddb2 -la detailed
```

Example 3

To analyze the last two days of information from the IBMSNAP_APPLYTRAIL, IBMSNAP_APPLYTRACE, IBMSNAP_CAPTRACE, IBMSNAP_CAPMON, and IBMSNAP_SIGNAL tables on the proddb1 and proddb2 databases:

```
asnanalyze -db proddb1 proddb2 -tl 2 -at 2 -ct 2 -cm 2 -sg 2
```

Example 4

To obtain a simple level of analysis about the last two days of information from the IBMSNAP_APPLYTRAIL, IBMSNAP_APPLYTRACE, IBMSNAP_CAPTRACE, IBMSNAP_CAPMON, and IBMSNAP_SIGNAL tables on the proddb1 and proddb2 databases for only the qual1 and qual2 Apply qualifiers:

```
asnanalyze -db proddb1 proddb2 -la simple -tl 2 -at 2 -ct 2 -cm 2 -sg 2
  -aq qual1 qual2 -od c:\mydir -fn anzout -pw c:\SQLLIB
```

This command example writes the analyzer output to a file named anzout under the c:\mydir directory and uses the password information from the c:\SQLLIB directory.

Example 5

To analyze a specific Capture schema:

```
asnanalyze -db proddb1 proddb2 -cs BSN
```

Example 6

To display command help:

```
asnanalyze
```

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.

Command help appears if you enter the **asnpwd** command without any parameters, followed by a *?*, or followed by incorrect parameters.

Syntax

```
►►──asnpwd──┬─init──┤ Init parameters ├───┬──────────────────────────►◄
            ├─add──┤ Add parameters ├─────┤
            ├─modify──┤ Modify parameters ├┤
            ├─delete──┤ Delete parameters ├┤
            └─list──┤ List parameters ├───┘
```

Init parameters:

```
├──┬─────────────────────────┬──┬──────────────────────────────┬──────────────┤
   └─encrypt──┬─all──────┬───┘  └─using──┬─asnpwd.aut─────┬─┘
              └─password─┘               └─filepath_name──┘
```

Add parameters:

```
├──alias──db_alias──id──userid──password──password──────────────────►
```

```
►─┬─────────────────────────┬──────────────────────────────────────────────┤
  │          ┌─asnpwd.aut────┐ │
  └─using───┴─filepath_name─┴─┘
```

Modify parameters:

```
├──alias──db_alias──id──userid──password──password──────────────────────────►

►─┬─────────────────────────┬──────────────────────────────────────────────┤
  │          ┌─asnpwd.aut────┐ │
  └─using───┴─filepath_name─┴─┘
```

Delete parameters:

```
├──alias──db_alias──┬─────────────────────────┬────────────────────────────┤
                    │          ┌─asnpwd.aut────┐ │
                    └─using───┴─filepath_name─┴─┘
```

List parameters:

```
├──┬─────────────────────────┬──────────────────────────────────────────────┤
   │          ┌─asnpwd.aut────┐ │
   └─using───┴─filepath_name─┴─┘
```

Parameters

Table 26 defines the invocation parameters for the **asnpwd** command.

Important note about compatibility of password files: Password files that are created by the **asnpwd** command starting with Version 9.5 Fix Pack 2 use a new encryption method and cannot be read by older versions of the replication programs and utilities. If you share a password file among programs and utilities that are at mixed level, with some older than these fix packs, do not recreate the password file by using an **asnpwd** utility that is at these fix packs or newer. Replication programs and utilities at these fix packs or newer can continue to work with older password files. Also, you cannot change an older password file to use the new encryption method; you must create a new password file.

Usage note: On 64-bit Windows operating systems, the ADD, MODIFY, DELETE, and LIST options are not supported for password files that were created by using the **asnpwd** command before Version 9.5 Fix Pack 2.

Table 26. asnpwd invocation parameter definitions for Linux, UNIX, and Windows operating systems

Parameter	Definition
init	Specify to create an empty password file. This command will fail if you specify the **init** parameter with a password file that already exists.
add	Specify to add an entry to the password file. There can only be one entry in the password file per *db_alias*. This command will fail if you specify the **add** parameter with an entry that already exists in the password file. Use the **modify** parameter to change an existing entry in the password file.

Table 26. asnpwd invocation parameter definitions for Linux, UNIX, and Windows operating systems (continued)

Parameter	Definition
modify	Specify to modify the password or user ID for an entry in the password file.
delete	Specify to delete an entry from the password file.
list	Specify to list the aliases and user ID entries in a password file. This parameter can be used only if the password file was created by using the **encrypt password** parameter. Passwords are never displayed by the **list** command.
encrypt	Specifies which entries in a file to encrypt. **all (default)** Encrypt all entries in the specified file such that you cannot list the database aliases, user names, and passwords that are in the file. This option reduces the exposure of information in password files. **password** Encrypt the password entry in the specified file. This option allows users to list the database aliases and user names stored in their password file. Passwords can never be displayed.
using *filepath*	Specifies the path and name of the password file. Follow the file naming conventions of your operating system. An example of a valid password file on Windows is `C:\sqllib\mypwd.aut`. If you specify the path and name of the password file, the path and the password file must already exist. If you are using the **init** parameter and you specify the path and name of the password file, the path must already exist and the command will create the password file for you. If you do not specify this parameter, the default file name is `asnpwd.aut` and the default file path is the current directory.
alias *db_alias*	Specifies the alias of the database to which the user ID has access. The alias is always folded to uppercase, regardless of how it is entered.
id *userid*	Specifies the user ID that has access to the database.
password *password*	Specifies the password for the specified user ID. This password is case sensitive and is encrypted in the password file.

Return Codes

The **asnpwd** command returns a zero return code upon successful completion. A nonzero return code is returned if the command is unsuccessful.

Examples for asnpwd

The following examples illustrate how to use the **asnpwd** command.

Example 1

To create a password file with the default name of `asnpwd.aut` in the current directory:

```
asnpwd INIT
```

Example 2

To create a password file named `pass1.aut` in the `c:\myfiles` directory:

```
asnpwd INIT USING c:\myfiles\pass1.aut
```

Example 3

To create a password file named `mypwd.aut` with the **`encrypt all`** parameter:

```
asnpwd INIT ENCRYPT ALL USING mypwd.aut
```

Example 4

To create a password file named `mypwd.aut` with the **`encrypt password`** parameter:

```
asnpwd INIT ENCRYPT PASSWORD USING mypwd.aut
```

Example 5

To create a default password file with 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 by 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 by using the **encrypt all** parameter, the following details are listed:

  ```
  asnpwd LIST USING pass1.aut
  Alias: SAMPLE  ID: chglocalid
  Number of Entries: 1
  ```

asnscrt: Creating a replication service

Use the **asnscrt** command to create a replication service in the Windows Service Control Manager (SCM) and invoke the **asnqcap**, **asnqapp**, **asnmon**, **asncap**, and **asnapply** commands. Run the **asnscrt** command on the Windows operating system.

Syntax

```
►►──asnscrt───┬─-QC─┬──db2_instance──account──password───┬─asnqcap_command──┬──────────►◄
              ├─-QA─┤                                    ├─asnqapp_command──┤
              ├─-M──┤                                    ├─asnmon_command───┤
              ├─-C──┤                                    ├─asncap_command───┤
              └─-A──┘                                    └─asnapply_command─┘
```

Parameters

Table 27 defines the invocation parameters for the **asnscrt** command.

Table 27. asnscrt invocation parameter definitions for Windows operating systems

Parameter	Definition
-QC	Specifies that you are starting a Q Capture program.
-QA	Specifies that you are starting a Q Apply program.
-M	Specifies that you are starting a Replication Alert Monitor program.
-C	Specifies that you are starting a Capture program.
-A	Specifies that you are starting an Apply program.
db2_instance	Specifies the DB2 instance used to identify a unique DB2 replication service. The DB2 instance can be a maximum of eight characters.
account	Specifies the account name that you use to log on to Windows. If the account is local it must begin with a period and a backslash (`.\`). Otherwise the domain or machine name must be specified (for example, `domain_name\account_name`).

Table 27. asnscrt invocation parameter definitions for Windows operating systems (continued)

Parameter	Definition
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. If the DB2PATH environment variable is not defined, you must specify a location for the work files by including the **capture_path** parameter with the **asnqcap** command. If the DB2PATH variable is defined and you specify a **capture_path**, the **capture_path** parameter overrides the DB2PATH variable. The **asnscrt** command does not validate the syntax of the **asnqcap** parameters that you enter.
asnqapp_command	Specifies the complete **asnqapp** command to start a Q apply program. Use the documented **asnqapp** command syntax with the appropriate **asnqapp** parameters. If the DB2PATH environment variable is not defined, you must specify the location for the work files by including the **apply_path** parameter with the **asnqapp** command. If the DB2PATH variable is defined and you specify an **apply_path**, the **apply_path** parameter overrides the DB2PATH variable. The **asnscrt** command does not validate the syntax of the **asnqapp** parameters that you enter.
asnmon_command	Specifies the complete **asnmon** command to start a Replication Alert Monitor program. Use the documented **asnmon** command syntax with the appropriate **asnmon** parameters. If the DB2PATH environment variable is not defined, you must specify a location for the log files by including the **monitor_path** parameter with the **asnmon** command. If the DB2PATH variable is defined and you specify a **monitor_path**, the **monitor_path** parameter overrides the DB2PATH variable. The **asnscrt** command does not validate the syntax of the **asnmon** parameters that you enter.
asncap_command	Specifies the complete **asncap** command to start a Capture program. Use the documented **asncap** command syntax with the appropriate **asncap** parameters. If the DB2PATH environment variable is not defined, you must specify a location for the work files by including the **capture_path** parameter with the **asncap** command. If the DB2PATH variable is defined and you specify a **capture_path**, the **capture_path** parameter overrides the DB2PATH variable. The **asnscrt** command does not validate the syntax of the **asncap** parameters that you enter.

Table 27. asnscrt invocation parameter definitions for Windows operating systems (continued)

Parameter	Definition
asnapply_command	Specifies the complete **asnapply** command to start an Apply program. Use the documented **asnapply** command syntax with the appropriate **asnapply** parameters. If the DB2PATH environment variable is not defined, you must specify the location for the work files by including the **apply_path** parameter with the **asnapply** command. If the DB2PATH variable is defined and you specify an **apply_path**, the **apply_path** parameter overrides the DB2PATH variable. The **asnscrt** command does not validate the syntax of the **asnapply** parameters that you enter.

Examples for asnscrt

The following examples illustrate how to use the **asnscrt** command.

Example 1

To create a DB2 replication service that invokes a Q Apply program under a DB2 instance called inst2 and uses a logon account of .\joesmith and a password of my$pwd:

```
asnscrt -QA inst2 .\joesmith my\$pwd asnqapp apply_server=mydb2 apply_schema =as2
  apply_path=X:\sqllib
```

Example 2

To create a DB2 replication service that invokes a Capture program under a DB2 instance called inst1:

```
asnscrt -C inst1 .\joesmith password asncap capture_server=sampledb
  capture_schema=ASN capture_path=X:\logfiles
```

Example 3

To create a DB2 replication service that invokes an Apply program under a DB2 instance called inst2 and uses a logon account of .\joesmith and a password of my$pwd:

```
asnscrt -A inst2 .\joesmith my\$pwd asnapply control_server=db2 apply_qual=aq2
  apply_path=X:\sqllib
```

Example 4

To create a DB2 replication service that invokes a Replication Alert Monitor program under a DB2 instance called inst3:

```
asnscrt -M inst3 .\joesmith password asnmon monitor_server=db3 monitor_qual=mq3
  monitor_path=X:\logfiles
```

Example 5

To create a DB2 replication service that invokes a Capture program under a DB2 instance called inst4 and overrides the default work file directory with a fully qualified **capture_path**:

```
asnscrt -C inst4 .\joesmith password X:\sqllib\bin\asncap capture_server=scdb
  capture_schema=ASN capture_path=X:\logfiles
```

Example 6

To create a DB2 replication service that invokes a Q capture program under a DB2 instance called inst1:

```
asnscrt -QC inst1 .\joesmith password asnqcap capture_server=mydb1
  capture_schema=QC1 capture_path=X:\logfiles
```

asnsdrop: Dropping a replication service

Use the **asnsdrop** command to drop replication services from the Windows Service Control Manager (SCM) on the Windows operating system.

Syntax

```
►►──asnsdrop──┬─service_name─┬──────────────────────────────────────────────►◄
              └─ALL──────────┘
```

Parameters

Table 28 defines the invocation parameters for the **asnsdrop** command.

Table 28. 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 29 defines the invocation parameter for the **asnslist** command.

Table 29. asnslist invocation parameter definition for Windows operating systems

Parameter	Definition
details	Specifies that you want to list detailed data about all DB2 replication services on a system.

Examples for asnlist

The following examples illustrate how to use the **asnslist** command.

Example 1

To list the names of DB2 replication services on a system:

```
asnslist
```

Here is an example of the command output:

```
DB2.DB2.SAMPLE.QAPP.ASN
DB2.DB4.SAMPLE.QCAP.ASN
```

Example 2

To list details about all services on a system:

```
asnslist details
```

Here is an example of the command output:

```
DB2.DB2.SAMPLE.QAPP.ASN
Display Name: DB2 DB2 SAMPLE QAPPLY ASN
Image Path:   ASNSERV DB2.DB2.SAMPLE.APP.AQ1 -ASNQAPPLY QAPPLY_SERVER=SAMPLE AP
              PLY_SCHEMA=ASN QAPPLY_PATH=C:\PROGRA~1\SQLLIB
Dependency:   DB2-0

DB2.DB4.SAMPLE.QCAP.ASN
Display Name: DB2 DB4 SAMPLE QAPPLY ASN
Image Path:   ASNSERV DB2.DB4.SAMPLE.APP.AQ1 -ASNQCAP QCAPTURE_SERVER=SAMPLE CA
              PTURE_SCHEMA=ASN QCAPTURE_PATH=C:\PROGRA~1\SQLLIB
Dependency:   DB4-0
```

asntdiff: Comparing data in source and target tables (Linux, UNIX, Windows)

Use the **asntdiff** command to compare two relational tables and generate a list of differences between the two. Run the **asntdiff** command at an operating system prompt or in a shell script.

This topic describes usage on Linux, UNIX, or Windows. For details on running **asntdiff** on z/OS, see “asntdiff: Comparing data in source and target tables (z/OS)” on page 274. For information on the asntdiff –f command option, which enables you to compare tables whether or not they are involved in replication by using an input file, see “asntdiff –f (input file) command option” on page 281.

The tables that you compare can reside on DB2 for Linux, UNIX, Windows, DB2 for z/OS, or DB2 for System i.

Syntax

```
►►──asntdiff──DB=server──┬─────────────────┬─────────────────────────────────►
                         └─SCHEMA=schema───┘

►─┬──────────────────────────────────────────┬─┬──────────────────────────────┬─►
  └─DIFF_SCHEMA=difference_table_schema──────┘ └─DIFF_TABLESPACE=tablespace───┘

►─┬─────────────────────────┬─┬──────────────────────────────┬─WHERE=WHERE_clause──►
  │                ┌─n─┐    │ └─MAXDIFF=difference_limit─────┘
  └─DIFF_DROP=─────┴─y─┴────┘

►─┬──────────────────────┬─┬───────────────────┬─┬──────────────────────┬───────►
  └─DIFF_PATH=log_path───┘ └─PWDFILE=filename──┘ └─DIFF=table_name──────┘

►─┬───────────────────────────────────────────┬──────────────────────────────►◄
  └─RANGECOL=─┤ range_clause_option ├─────────┘
```

range_clause_option:

```
├──┬─src_colname─FROM:date-time_lower-bound─TO:date-time_upper-bound─┬──────────┤
   ├─src_colname─FROM:date-time──────────────────────────────────────┤
   └─src_colname─TO:date-time────────────────────────────────────────┘
```

Parameters

Table 30 defines the invocation parameters for the **asntdiff** command.

Table 30. asntdiff invocation parameter definitions for Linux, UNIX, and Windows operating systems

Parameter	Definition
DB=*server*	Specifies the DB2 alias of the database that stores information about the source and target tables to be compared. The value differs depending on whether you are using Q Replication or SQL Replication: **Q Replication** The name of the Q Capture server, which contains the IBMQREP_SUBS table. **SQL Replication** The name of the Apply control server, which contains the IBMSNAP_SUBS_MEMBR table.
SCHEMA=*schema*	Specifies the schema of the Q Capture control tables for Q Replication, or the schema of the Apply control tables for SQL Replication. The default is ASN.
DIFF_SCHEMA= *difference_table_schema*	Specifies the schema of the difference table. The default is ASN.
DIFF_TABLESPACE=*tablespace*	Specifies the table space of the difference table. If this parameter is not specified, the table is created in the default table space in the database where the **asntdiff** command was run.
DIFF_DROP=y/n	Specifies whether an existing difference table will be dropped and recreated before it is reused to record differences. If the table does not exist, the **asntdiff** command creates it. **n (default)** The difference table will be used as is and the existing rows will be deleted. **y** The difference table will be dropped and recreated.
MAXDIFF=*difference_limit*	Specifies the maximum number of differences that you want the **asntdiff** command to process before it stops. The default value is 10000.

Table 30. asntdiff invocation parameter definitions for Linux, UNIX, and Windows operating systems (continued)

Parameter	Definition
WHERE=*WHERE_clause*	Specifies an SQL WHERE clause that uniquely identifies one row of the control table that stores information about the source and target tables that will be compared. The WHERE clause must be in double quotation marks. The value of this parameter differs depending on whether you are using Q Replication or SQL Replication: **Q Replication** The WHERE clause specifies a row in the IBMQREP_SUBS table and uses the SUBNAME column to identify the Q subscription that contains the source and target tables. **SQL Replication** The WHERE clause specifies a row in the IBMSNAP_SUBS_MEMBR table and uses the SET_NAME, APPLY_QUAL, TARGET_SCHEMA, and TARGET_TABLE columns to identify the subscription set member that contains the source and target tables.
DIFF_PATH=*log_path*	Specifies the location where you want the asntdiff command to write its log. The default value is the directory from which you ran the command. The value must be an absolute path name. Use double quotation marks ("") to preserve case.
PWDFILE=*filename*	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 `asnpwd.aut` (the name of the password file that is created by the **asnpwd** command). The **asntdiff** 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.
DIFF=*table_name*	Specifies the name of the table that is created in the source database to store differences between the source and target tables. The table has one row for each difference that is detected. If you do not include this parameter or the **DIFF_SCHEMA** parameter, the difference table is named ASN.ASNTDIFF.

Table 30. asntdiff invocation parameter definitions for Linux, UNIX, and Windows operating systems (continued)

Parameter	Definition

`RANGECOL` clause	Specifies a range of rows from the source table that you want to compare. You provide the name of a DATE, TIME, or TIMESTAMP column in the source table, and then use one of three different clauses for specifying the range. The column name must be enclosed in single quotation marks. The clause must be enclosed in double quotation marks.
	
	The timestamp uses the following format: *YYYY-MM-DD-HH.MM.SS.mmmmm*. For example, 2010-03-10-10.35.30.55555 is the GMT timestamp for March 10, 2010, 10:35 AM, 30 seconds, and 55555 microseconds.
	
	Use one of the following clauses:
	
	src_colname **FROM:** *date-time_lower-bound* **TO:** *date-time_upper-bound*
	Specifies a lower and upper bound for the range of rows to compare.
	
	The following example uses a TIMESTAMP column:
	`"'SALETIME'`
	`FROM: 2008-02-08-03.00.00.00000`
	`TO: 2008-02-15-03.00.00.00000"`
	
	Remember: Both the **`FROM:`** and **`TO:`** keywords are required and both keywords must be followed by a colon (:).
	
	src_colname **FROM:** *date-time*
	Specifies that you want to compare all rows with timestamps that are greater than or equal to *date-time*.
	
	For example:
	`"'SALE_TIME'`
	`FROM: 2008-03-10-10.35.30.55555"`
	
	src_colname **TO:** *date-time*
	Specifies that you want to compare all rows with timestamps that are less than or equal to the *date-time*.
	
	For example:
	`"'SALETIME'`
	`TO: 2008-03-20-12.00.00.00000"`
	Recommendation: For better performance, ensure that you have an index on the source column that is specified in the range clause.When you compare tables that are involved in peer-to-peer replication, you can use the IBM-generated IBMQREPVERTIME column for the source column in the range clause.
	Restriction: The RANGECOL parameter is not valid for the **`asntdiff -f`** (input file) option. You can use a SQL WHERE clause in the input file to achieve similar results.

Examples for asntdiff

The following examples show how to use the **asntdiff** command.

Example 1

In Q Replication, to find the differences between a source and target table that are specified in a Q subscription named my_qsub, on a Q Capture server named source_db, with a Q Capture schema of asn:

```
asntdiff db=source_db schema=asn where="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
```

Example 3

In Q Replication, to find the differences between a range of rows in the source and target tables that are specified in a peer-to-peer 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'"
 RANGECOL="'IBMQREPVERTIME' FROM: '2008-03-10-0.00.00.00000'
 TO: '2007-04-12-00.00.00.00000'"
```

Example 4

In SQL Replication, to find the differences between a range of rows in the 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
 RANGECOL="'CREDIT_TIME' FROM:'2008-03-10-12.00.00.00000'
 TO: '2008-03-11-12.00.00.00000'"
```

asntdiff: Comparing data in source and target tables (z/OS)

Use the **asntdiff** command to compare two relational tables and generate a list of differences between the two. Run the **asntdiff** command with JCL or at a UNIX System Services (USS) command prompt or shell script.

This topic describes usage on z/OS:

- For details on running **asntdiff** on Linux, UNIX, and Windows, see “asntdiff: Comparing data in source and target tables (Linux, UNIX, Windows)” on page 270.
- For information on the asntdiff –f command option, which enables you to compare tables whether or not they are involved in replication, see “asntdiff –f (input file) command option” on page 281.
- For details on using asntdiff in parallel mode, see “Running the asntdiff utility in parallel mode (z/OS)” on page 210.

The tables that you compare can reside on DB2 for z/OS, DB2 for Linux, UNIX, Windows, or DB2 for System i.

Syntax

```
►►──asntdiff──DB=server──DB2_SUBSYSTEM=subsystem──┬──────────────────┬──────────►
                                                   └─SCHEMA=schema──┘

►─┬──────────────────────────────────────────┬─┬─────────────────────────────┬──►
  └─DIFF_SCHEMA=difference_table_schema──────┘ └─DIFF_TABLESPACE=tablespace─┘

►─┬────────────────────┬─┬──────────────────────────────┬─WHERE=WHERE_clause─►
  │              ┌─n─┐ │ └─MAXDIFF=difference_limit────┘
  └─DIFF_DROP=───┴─y─┴─┘

►─┬──────────────────┬─┬─────────────────────────────────────┬──────────────────►
  └─DIFF=table_name──┘ └─RANGECOL=─┤ range-clause-option ├──┘

►─┬───────────────────────────────────────────────┬──────────────────────────────►
  │           ┌─n──────────────────────────────┐ │
  └─PARALLEL=─┴─y─┤ parallel-options ├─────────┴─┘

►─┬──────────────────────────────────────────────┬──────────────────────────────►◄
  ├─SQLID=source_authorization_ID────────────────┤
  └─SOURCESQLID=source_authorization_ID──────────┘
```

range-clause-option:

```
├──┬─src_colname─FROM:date-time_lower-bound─TO:date-time_upper-bound─┬──────────┤
   ├─src_colname─FROM:date-time──────────────────────────────────────┤
   └─src_colname─TO:date-time────────────────────────────────────────┘
```

parallel-options:

```
├─┬──────────────────────────────────────┬─┬──────────────────────────────┬─┬──────────────────────────────────────────┬─┤
  └─NUMTHREADS=level_of_parallelism──────┘ └─NUMBLOCKS==number_of_blocks──┘ └─TARGET_SQLID=target_authorization_ID───┘
```

Parameters

Table 31 defines the invocation parameters for the **asntdiff** command.

Table 31. asntdiff invocation parameter definitions for z/OS operating systems

Parameter	Definition
DB=*server*	Specifies the DB2 alias of the database that stores information about the source and target tables to be compared. The value differs depending on whether you are using Q Replication or SQL Replication: **Q Replication** The location name of the Q Capture server, which contains the IBMQREP_SUBS table. **SQL Replication** The location name of the Apply control server, which contains the IBMSNAP_SUBS_MEMBR table. **Restriction:** This parameter is not valid for the asntdiff -f (input file) option. You can use the SOURCE_SERVER and TARGET_SERVER parameters with the -f option to specify the names of the source and target database. On z/OS, these are location names and you also must use the DB2_SUBSYSTEM parameter to specify the name of the subsystem where the asntdiff utility runs.
DB2_SUBSYSTEM=*subsystem*	Specifies the name of the subsystem where you run the asntdiff command.
SCHEMA=*schema*	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 ASN.
DIFF_SCHEMA= *difference_table_schema*	Specifies the schema of the difference table. The default is ASN.
DIFF_TABLESPACE=*tablespace*	Specifies the table space of the difference table. If this parameter is not specified, the table is created in the default table space in the subsystem where the **asntdiff** command was run. This is a two-part name, *dbname.tablespace*, where *dbname* is the logical database name and *tablespace* is the table space name.
DIFF_DROP=y/n	Specifies whether an existing difference table will be dropped and recreated before it is reused to record differences. If the table does not exist, the **asntdiff** command creates it. **n (default)** The difference table will be used as is and the existing rows will be deleted. **y** The difference table will be dropped and recreated.
MAXDIFF=*difference_limit*	Specifies the maximum number of differences that you want the **asntdiff** command to process before it stops. The default value is 10000.

Table 31. asntdiff invocation parameter definitions for z/OS operating systems (continued)

Parameter	Definition
WHERE=*WHERE_clause*	Specifies an SQL WHERE clause that uniquely identifies one row of the control table that stores information about the source and target tables that will be compared. The WHERE clause must be in double quotation marks. The value of this parameter differs depending on whether you are using Q Replication or SQL Replication: **Q Replication** The WHERE clause specifies a row in the IBMQREP_SUBS table and uses the SUBNAME column to identify the Q subscription that contains the source and target tables. **SQL Replication** The WHERE clause specifies a row in the IBMSNAP_SUBS_MEMBR table and uses the SET_NAME, APPLY_QUAL, TARGET_SCHEMA, and TARGET_TABLE columns to identify the subscription set member that contains the source and target tables. **Restriction:** This parameter is not valid for the asntdiff -f (input file) option. You can use the SOURCE_SELECT and TARGET_SELECT parameters with the -f option to specify the tables to be compared, and can use a WHERE clause in the SQL query that is provided with these parameters.
DIFF=*table_name*	Specifies the name of the table that is created in the source subsystem to store differences between the source and target tables. The table has one row for each difference that is detected. If you do not include this parameter or the **DIFF_SCHEMA** parameter, the difference table is named ASN.ASNTDIFF.

Table 31. asntdiff invocation parameter definitions for z/OS operating systems (continued)

Parameter	Definition

RANGECOL clause	Specifies a range of rows from the source table that you want to compare. You provide the name of a DATE, TIME, or TIMESTAMP column in the source table, and then use one of three different clauses for specifying the range. The column name must be enclosed in single quotation marks. The clause must be enclosed in double quotation marks.
	
	The timestamp uses the following format: *YYYY-MM-DD-HH.MM.SS.mmmmm*. For example, 2010-03-10-10.35.30.55555 is the GMT timestamp for March 10, 2010, 10:35 AM, 30 seconds, and 55555 microseconds.
	
	Use one of the following clauses:
	
	src_colname **FROM:** *date-time_lower-bound* **TO:** *date-time_upper-bound*
	Specifies a lower and upper bound for the range of rows to compare.
	
	The following example uses a TIMESTAMP column:
	`"'SALETIME'`
	`FROM: 2008-02-08-03.00.00.00000`
	`TO: 2008-02-15-03.00.00.00000"`
	
	Remember: Both the **FROM:** and **TO:** keywords are required and both keywords must be followed by a colon (:).
	
	src_colname **FROM:** *date-time*
	Specifies that you want to compare all rows with timestamps that are greater than or equal to *date-time*.
	
	For example:
	`"'SALE_TIME'`
	`FROM: 2008-03-10-10.35.30.55555"`
	
	src_colname **TO:** *date-time*
	Specifies that you want to compare all rows with timestamps that are less than or equal to the *date-time*.
	
	For example:
	`"'SALETIME'`
	`TO: 2008-03-20-12.00.00.00000"`
	Recommendation: For better performance, ensure that you have an index on the source column that is specified in the range clause.When you compare tables that are involved in peer-to-peer replication, you can use the IBM-generated IBMQREPVERTIME column for the source column in the range clause.
	Restriction: The RANGECOL parameter is not valid for the **asntdiff -f** (input file) option. You can use a SQL WHERE clause in the input file to achieve similar results.

Table 31. asntdiff invocation parameter definitions for z/OS operating systems (continued)

Parameter	Definition
PARALLEL=y/n	Specifies whether the asntdiff utility uses parallel mode, in which multiple threads are used to compare the tables, or operates in serial mode with a single thread. **n (default)** The asntdiff utility uses serial mode. **y** The asntdiff utility uses parallel mode. For details on installation requirements, required authorizations, and restrictions, see "Running the asntdiff utility in parallel mode (z/OS)" on page 210.
NUMTHREADS=*number_of_threads*	Specifies the number of threads that the asntdiff utility is allowed to create. The minimum value is six. The recommended value is 21, which is also the maximum value and the default value. Ensure that the MAXTHREADS parameter value that is specified in BPXPRMXX is larger than the specified number of threads. Also, configure DB2 ZPARMS CTHREAD, IDFORE, and IDBACK to allow each of the created threads to connect to DB2.
NUMBLOCKS=*number_of_blocks*	Specifies the number of partitions into which the asntdiff utility divides the source and target tables (that is, the result sets of the SOURCE_SELECT and TARGET_SELECT parameters) for parallel compare. A value of 0 (the default) means that the utility automatically determines the number of blocks.
SQLID=*authorization_ID*	Use this parameter when asntdiff is running in non-parallel mode. The parameter specifies an authorization ID that can be used to create the difference table. Use this parameter if the ID that is used to run the **asntdiff** command does not have authorization to create tables. The value of the **SQLID** parameter is used as the schema for the difference table if you do not explicitly specify a schema by using the **DIFF_SCHEMA** parameter.
SOURCE_SQLID=*authorization_ID*	When you use asntdiff in parallel mode, this parameter specifies an authorization ID that can be used to execute stored procedures and packages and run DDL and DML on temporary tables at the source. Use this parameter if the ID that is used to run the **asntdiff** command does not have the necessary authorization.
TARGET_SQLID=*authorization_ID*	When you use asntdiff in parallel mode, this parameter specifies an authorization ID that can be used to execute stored procedures and packages and run DDL and DML on temporary tables at the target. Use this parameter if the ID that is used to run the **asntdiff** command does not have the necessary authorization.

Usage notes

The **asntdiff** command creates data sets (JCL) or temporary files (USS) for spilling data and for writing differences before inserting them into the difference table. You specify the location of the data sets or temporary files differently:

JCL

If you want ASNTDIFF to write to z/OS data sets, add these two DD statements to your ASNTDIFF JCL, modifying the size specifications to match the size of your source table:

```
//SPLFILE  DD  DSN=&&SPILL,DISP=(NEW,DELETE,DELETE),
//             UNIT=VIO,SPACE=(CYL,(11,7)),
//             DCB=(RECFM=VS,BLKSIZE=6404)
//DIFFFILE DD  DSN=&&DIFFLE,DISP=(NEW,DELETE,DELETE),
//             UNIT=VIO,SPACE=(CYL,(11,7)),
//             DCB=(RECFM=VS,BLKSIZE=6404)
```

USS On USS, temporary files are written by default to the hierarchical file system (HFS), in the home directory of the user ID that executes the **asntdiff** command. The default names are `DD:DIFFFILE` and `DD:SPILLFILE`. You can use a DIFFFILE DD statement to specify an alternative HFS path and file name for those files, as shown in this example:

```
//DIFFFILE DD PATH='/u/oeusr01/tdiffil2',
//         PATHDISP=(KEEP,KEEP),
//         PATHOPTS=(ORDWR,OCREAT),
//         PATHMODE=(SIRWXU,SIRGRP,SIROTH)
```

Redirecting the HFS requires you to create an empty file that can be written to or to use the above PATHDISP and PATHOPTS settings to create a new file if one does not exist.

Examples for asntdiff

The first four examples show how to use the **asntdiff** command on USS; the fifth example provides JCL. For more sample JCL, see the ASNTDIFF sample program in the SASNSAMP sample data set.

Example 1: Q Replication

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: SQL Replication

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
```

Example 3: Comparing a range of rows in Q Replication

In Q Replication, to find the differences between a range of rows in the source and target tables that are specified in a peer-to-peer 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'"
 RANGECOL="'IBMQREPVERTIME' FROM: '2008-03-10-0.00.00.00000'
 TO: '2007-04-12-00.00.00.00000'"
```

Example 4: Comparing a range of rows in SQL Replication

In SQL Replication, to find the differences between a range of rows in the 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
 RANGECOL="'CREDIT_TIME' FROM:'2008-03-10-12.00.00.00000'
 TO: '2008-03-11-12.00.00.00000'"
```

Example 5: Using asntdiff in parallel mode

To run the asntdiff utility in parallel mode to compare two tables with 21 parallel threads, you can use the following JCL after locating and changing all occurrences of the following strings:

- The subsystem name DSN! to the name of your DB2 subsystem
- DSN!!0 to the name if your DB2 target library
- ASNQ!!0 to the name of your Replication Server target library

```
//ASNTDIF1 EXEC PGM=ASNTDIFF,PARM='/-F'
//STEPLIB  DD DSN=ASNQ!!0.SASNLOAD,DISP=SHR
//            DD DSN=DSN!!0.SDSNLOAD,DISP=SHR
//MSGS     DD PATH='/usr/lpp/db2repl_10_01/msg/En_US/db2asn.cat'
//CEEDUMP  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  DUMMY
//SYSIN    DD   *
DB2_SUBSYSTEM=DSN!
SOURCE_SERVER=DQRG
SOURCE_SELECT="SELECT empno, department FROM employee
                            WHERE empno > 10000 ORDER BY 1"
TARGET_SERVER=D7DP
TARGET_SELECT="SELECT empno, department FROM employee
                            WHERE empno > 10000 ORDER BY 1"
PARALLEL=Y
NUMTHREADS=21
SOURCE_SQLID=SRCADM
TARGET_SQLID=TGTADM
DIFF_DROP=Y
MAXDIFF=20000
DEBUG=NO
/*
//
```

asntdiff –f (input file) command option

With the **asntdiff -f** command option, you use an input file to specify information about any two tables that you want to compare, whether or not they are being replicated.

The input file contains SQL SELECT statements for the source and target tables that specify the rows that you want to compare. The standard asntdiff command compares tables that are involved in replication by using subscription information from the replication control tables.

The **asntdiff -f** option can compare any tables on z/OS, Linux, UNIX, or Windows. You can run **asntdiff -f** from a Linux, UNIX, or Windows command prompt, from z/OS as a batch job that uses JCL, or from z/OS under the UNIX System Services (USS) environment.

In addition to the SELECT statements, the input file contains the source and target database information, the difference table information, and optional parameters that specify methods for processing the differences. You can use a password file that is created by the **asnpwd** command to specify a user ID and password for connecting to the source and target databases.

Note: The asntrep command for repairing table differences does not support the input file option.

The format of the input file contents is as follows:

```
* Optional comment line
# Optional comment line
SOURCE_SERVER=server_name
SOURCE_SELECT="SQL_SELECT_STATEMENT"
TARGET_SERVER=server_name
TARGET_SELECT="SQL_SELECT_STATEMENT"
PARAMETER=value
...
```

Follow these guidelines:

- Each parameter must follow the *parameter=value* format.
- Multiple parameter-value pairs can be specified on a single line, separated by a blank. The parameter-value pairs also can be specified on a new line.
- To preserve blanks, surround parameter values with double quotation marks ("). Double quotation marks are also required for the source and target SELECT statements.
- If you want to preserve mixed case or blanks in the names of single DB2 objects (column or table names, DIFF_SCHEMA, DIFF_TABLESPACE) mask them with `\" \"`, for example `\"MY NAME\"` or `\"ColumnName\"` or `\"name\"`.
- Comments must be prefixed with an asterisk (*) or pound sign (#). This line is ignored. Comments must be on their own line and cannot be added to a line that contains parameters.
- Surround the DIFF_PATH and PWDFILE parameters with double quotation marks ("). A final path delimiter for DIFF_PATH is not required.

Syntax

```
▶▶──asntdiff──-f──input_filename──────────────────────────────────▶◀
```

Parameters

Table 32 defines the mandatory parameters to include in the input file for the **asntdiff -f** command.

For descriptions of optional parameters that you can include in the input file (and which are shared by the standard asntdiff command) see “asntdiff: Comparing data in source and target tables (z/OS)” on page 274 or“asntdiff: Comparing data in source and target tables (Linux, UNIX, Windows)” on page 270.

Table 32. asntdiff -f invocation parameter definitions for Linux, UNIX, Windows, and z/OS

Parameter	Definition
input_filename	Specifies the name of the file that contains the source and target database information and SELECT statements. Specify a directory path if the file is located somewhere other than the directory from which you run the **asntdiff -f** command.

Table 32. asntdiff -f invocation parameter definitions for Linux, UNIX, Windows, and z/OS (continued)

Parameter	Definition
SOURCE_SERVER= *source_server_name*	Specifies the alias of the database where the source table exists.
TARGET_SERVER= *target_server_name*	Specifies the alias of the database where the target table exists.
SOURCE_SELECT= *source_select_statement* **TARGET_SELECT**= *target_select_statement*	Any valid SQL SELECT statement. The result sets from the SQL statement at each table must contain columns with matching data types and lengths. The **asntdiff** command describes the queries and compares the data from the two result sets. The command does not explicitly check the system catalog for type and length information. The SELECT can be an open select as in (*), or a SELECT statement that contains column names, SQL expressions, and WHERE clauses that are permitted. An ORDER BY clause is mandatory. The clause must contain the numeric values of the positions of the columns in the SQL statement. Ensure that the column or columns in the ORDER BY clause reference a unique key or unique composite key. Otherwise the results are incorrect. An index on the columns in the ORDER BY clause might improve performance by eliminating the need for a sort. The entire statement must be enclosed in double quotes to mark the beginning and the end.

The following examples show the mandatory parameters, SQL statements, and optional parameters that you put in the input file.

z/OS

Example 1

This example shows the use of an open SELECT statement on DB2 for z/OS. Note the use of the \" to preserve mixed case in the table owner, and the use of optional parameters in the input file. Also note the use of the DB2_SUBSYSTEM parameter.

```
SOURCE_SERVER=STPLEX4A_DSN7
SOURCE_SELECT="select * from CXAIMS.ALDEC order by 1"
TARGET_SERVER=STPLEX4A_DSN7
TARGET_SELECT="select * from \"Cxaims\".TARG_ALDEC order by 1"
DIFF_DROP=Y
DB2_SUBSYSTEM=DSN7
MAXDIFF=10000
DEBUG=YES
```

z/OS

Example 2

This example demonstrates the use of SUBSTR and CAST functions in the SELECT statements.

```
SOURCE_SERVER=D7DP
SOURCE_SELECT="select HIST_CHAR12,HIST_DATE,HIST_CHAR6,HIST_INT1,HIST_INT2,
HIST_INT3,SUBSTR(CHAR1,1,5) AS CHAR1,SUBSTR(CHAR2,1,10) AS CHAR2,HIST_INT3,
HIST_DEC1,HIST_DEC2,HIST_DEC3,CAST(INT1 AS SMALLINT) AS INT1
FROM BISVT.THIST17 ORDER BY 4"
TARGET_SERVER=STPLEX4A_DSN7
TARGET_SELECT="select HIST_CHAR12,HIST_DATE,HIST_CHAR6,HIST_INT1,HIST_INT2,
HIST_INT3,CHAR1,CHAR2,HIST_INT3,HIST_DEC1,HIST_DEC2,HIST_DEC3,SML1
FROM BISVT.THIST17 ORDER BY 4"
DB2_SUBSYSTEM=DSN7
DIFF_DROP=Y
DEBUG=YES
MAXDIFF=10000
```

Windows

Example 3

This example compares the EMPLOYEE tables on SOURCEDB and TARGETDB and includes several optional parameters.

```
SOURCE_SERVER=SOURCEDB
SOURCE_SELECT="select FIRSTNME, LASTNAME, substr(WORKDEPT,1,1)
as WORKDEPT, EMPNO from EMPLOYEE order by 4"
TARGET_SERVER=TARGETDB
TARGET_SELECT="select FIRSTNME, LASTNAME, substr(WORKDEPT,1,1)
as WORKDEPT, EMPNO from EMPLOYEE order by 4"
DIFF_DROP=Y
DIFF =\"diffTable\"
DEBUG=YES
MAXDIFF=10000
PWDFILE="asnpwd.aut"
DIFF_PATH="C:\utils\"
```

Linux UNIX

Example 4

This example compares the EMPLOYEE tables in a Linux or UNIX environment and uses a casting function.

```
SOURCE_SERVER=SOURCEDB
SOURCE_SELECT="select EMPNO, FIRSTNME, LASTNAME, cast(SALARY as INT)
as SALARY from EMPLOYEE order by 1"
TARGET_SERVER=TARGETDB
TARGET_SELECT="select EMPNO, FIRSTNME, LASTNAME, cast(SALARY as INT)
as SALARY from EMPLOYEE order by 1"
DIFF_DROP=Y
DIFF =\"diffTable\"
DEBUG=YES
MAXDIFF=10000
PWDFILE="asnpwd.aut"
DIFF_PATH="home/laxmi/utils"
```

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.

You run this command at an operating system prompt or in a shell script.

Syntax

```
>>──asntrc──────────────────────────────────────────────────────────────────────>

>──┬─on──────-db──db_name──┬─qcap──┬──────────────────────────┬─┬─┤ On parameters ├────────┬──><
   │                       │       └─-schema──qcapture_schema─┘ │                              │
   │                       ├─qapp──┬────────────────────────┬───┤                              │
   │                       │       └─-schema──qapply_schema─┘   │                              │
   │                       ├─cap──┬─────────────────────────┬───┤                              │
   │                       │      └─-schema──capture_schema─┘   │                              │
   │                       ├─app──┬────────────────────────────┬┤                              │
   │                       │      └─-qualifier──apply_qualifier─┘│                              │
   │                       └─mon──┬──────────────────────────────┬┘                              │
   │                              └─-qualifier──monitor_qualifier─┘                              │
   ├─┬─off───────┬──-db──db_name──┬─qcap──┬──────────────────────────┬─┬─────────────────────────┤
   │ ├─kill──────┤                │       └─-schema──qcapture_schema─┘ │                         │
   │ ├─clr───────┤                ├─qapp──┬────────────────────────┬───┤                         │
   │ ├─diag──────┤                │       └─-schema──qapply_schema─┘   │                         │
   │ └─resetlock─┘                ├─cap──┬─────────────────────────┬───┤                         │
   │                              │      └─-schema──capture_schema─┘   │                         │
   │                              ├─app──┬────────────────────────────┬┤                         │
   │                              │      └─-qualifier──apply_qualifier─┘│                         │
   │                              └─mon──┬──────────────────────────────┬┘                         │
   │                                     └─-qualifier──monitor_qualifier─┘                         │
   ├─dmp──filename──-db──db_name──┬─qcap──┬──────────────────────────┬─┬─┬───────────┬─────────┤
   │                              │       └─-schema──qcapture_schema─┘ │ └─-holdlock─┘         │
   │                              ├─qapp──┬────────────────────────┬───┤                         │
   │                              │       └─-schema──qapply_schema─┘   │                         │
   │                              ├─cap──┬─────────────────────────┬───┤                         │
   │                              │      └─-schema──capture_schema─┘   │                         │
   │                              ├─app──┬────────────────────────────┬┤                         │
   │                              │      └─-qualifier──apply_qualifier─┘│                         │
   │                              └─mon──┬──────────────────────────────┬┘                         │
   │                                     └─-qualifier──monitor_qualifier─┘                         │
   ├─┬─flw───┬──┬──────────────────────────────────────────────────────┬─┤ Format parameters ├─┤
   │ ├─fmt───┤  └─┬───────────────────┬─┬─qcap──┬──────────────────────────┬─┤                    │
   │ └─v7fmt─┘    └─-db──db_name─┘      │       └─-schema──qcapture_schema─┘ │                    │
   │                                    ├─qapp──┬────────────────────────┬───┤                    │
   │                                    │       └─-schema──qapply_schema─┘   │                    │
   │                                    ├─cap──┬─────────────────────────┬───┤                    │
   │                                    │      └─-schema──capture_schema─┘   │                    │
   │                                    ├─app──┬────────────────────────────┬┤                    │
   │                                    │      └─-qualifier──apply_qualifier─┘│                    │
   │                                    └─mon──┬──────────────────────────────┬┘                    │
   │                                           └─-qualifier──monitor_qualifier─┘                    │
   ├─┬─stat─────┬──┬────────────────────────────────────────────────────────────┬──────────────┤
   │ └─statlong─┘  ├─┬───────────────────┬─┬─qcap──┬──────────────────────────┬─┤              │
   │               │ └─-db──db_name─┘      │       └─-schema──qcapture_schema─┘ │              │
   │               │                       ├─qapp──┬────────────────────────┬───┤              │
   │               │                       │       └─-schema──qapply_schema─┘   │              │
   │               │                       ├─cap──┬─────────────────────────┬───┤              │
   │               │                       │      └─-schema──capture_schema─┘   │              │
   │               │                       ├─app──┬────────────────────────────┬┤              │
   │               │                       │      └─-qualifier──apply_qualifier─┘│              │
   │               │                       └─mon──┬──────────────────────────────┬┘              │
   │               │                              └─-qualifier──monitor_qualifier─┘              │
   │               └─-fn──filename──────────────────────────────────────────────┘              │
   ├─-db──db_name──┬─qcap──┬──────────────────────────┬─┬─┤ Change settings parameters ├──────┤
   │               │       └─-schema──qcapture_schema─┘ │                                       │
   │               ├─qapp──┬────────────────────────┬───┤                                       │
   │               │       └─-schema──qapply_schema─┘   │                                       │
   │               ├─cap──┬─────────────────────────┬───┤                                       │
   │               │      └─-schema──capture_schema─┘   │                                       │
   │               ├─app──┬────────────────────────────┬┤                                       │
   │               │      └─-qualifier──apply_qualifier─┘│                                       │
   │               └─mon──┬──────────────────────────────┬┘                                       │
   │                      └─-qualifier──monitor_qualifier─┘                                       │
   ├─┬───────┬──────────────────────────────────────────────────────────────────────────────┤
   │ └─-help─┘                                                                              │
   └─-listsymbols───────────────────────────────────────────────────────────────────────────┘
```

On parameters:

```
├──┬───────────────────┬──┬──────────────┬──┬──────────────┬──────────────>
   └─-b──buffer_size─┘    └─-fn──filename─┘  └─-fs──filesize─┘

>──┬─────────────────┬──┬──────────────────────────────────────────────────┬──┤
   └─-d──diag_mask─┘    └─-df──function_name|component_name diag_mask─┘
```

Format parameters:

```
├──┬───────────────┬──┬──────────────┬──────────────────────────────────────►
   └─-fn─filename─┘  └─-d─diag_mask─┘

►──┬─────────────────────────────────────────────────┬──┬────────────┬──────┤
   └─-df─function_name|component_name diag_mask─┘  └─-holdlock─┘
```

Change settings parameters:

```
├──┬──────────────┬──┬─────────────────────────────────────────────────┬──────┤
   └─-d─diag_mask─┘  └─-df─function_name|component_name diag_mask─┘
```

Parameters

Table 33 defines the invocation parameters for the **asntrc** command.

Table 33. asntrc invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems

Parameter	Definition
on	Specify to turn on the trace facility for a specific Q Capture, Q Apply, Capture, Apply, or Replication Alert Monitor program. The trace facility creates a shared memory segment used during the tracing process.
-db *db_name*	Specifies the name of the database to be traced: • Specifies the name of the Q Capture server for the Q Capture program to be traced. • Specifies the name of the Q Apply server for the Q Apply program to be traced. • Specifies the name of the Capture control server for the Capture program to be traced. • Specifies the name of the Apply control server for the Apply program to be traced. • Specifies the name of the Monitor control server for the Replication Alert Monitor program to be traced.
-qcap	Specifies that a Q Capture program is to be traced. The Q Capture program is identified by the **-schema** parameter.
-schema *qcapture_schema*	Specifies the name of the Q Capture program to be traced. The Q Capture program is identified by the Q Capture schema that you enter. Use this parameter with the **-qcap** parameter.
-qapp	Specifies that a Q Apply program is to be traced. The Q Apply program is identified by the **-schema** parameter.
-schema *qapply_schema*	Specifies the name of the Q Apply program to be traced. The Q Apply program is identified by the Q Apply schema that you enter. Use this parameter with the **-qapp** parameter.
-cap	Specifies that a Capture program is to be traced. The Capture program is identified by the **-schema** parameter.

Table 33. asntrc invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
-schema *capture_schema*	Specifies the name of the Capture program to be traced. The Capture program is identified by the Capture schema that you enter. Use this parameter with the **-cap** parameter.
-app	Specifies that an Apply program is to be traced. The Apply program is identified by the **-qualifier** parameter.
-qualifier *apply_qualifier*	Specifies the name of Apply program to be traced. This Apply program is identified by the Apply qualifier that you enter. Use this parameter with the **-app** parameter.
-mon	Specifies that a Replication Alert Monitor program is to be traced. The Replication Alert Monitor program is identified by the **-qualifier** parameter.
-qualifier *monitor_qualifier*	Specifies the name of Replication Alert Monitor program to be traced. This Replication Alert Monitor program is identified by the monitor qualifier that you enter. Use this parameter with the **-mon** parameter.
off	Specify to turn off the trace facility for a specific Q Capture, Q Apply, Capture, Apply, or Replication Alert Monitor program and free the shared memory segment in use.
kill	Specify to force an abnormal termination of the trace facility. Use this parameter only if you encounter a problem and are unable to turn the trace facility off with the **off** parameter.
clr	Specify to clear a trace buffer. This parameter erases the contents of the trace buffer but leaves the buffer active.
diag	Specify to view the filter settings while the trace facility is running.
resetlock	Specify to release the buffer latch of a trace facility. This parameter enables the buffer latch to recover from an error condition in which the trace program terminated while holding the buffer latch.
dmp *filename*	Specify to write the current contents of the trace buffer to a file.
-holdlock	Specifies that the trace facility can complete a file dump or output command while holding a lock, even if the trace facility finds insufficient memory to copy the buffer.
flw	Specify to display summary information produced by the trace facility and stored in shared memory or in a file. This information includes the program flow and is displayed with indentations that show the function and call stack structures for each process and thread.

Table 33. asntrc invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
fmt	Specify to display detailed information produced by the trace facility and stored in shared memory or in a file. This parameter displays the entire contents of the traced data structures in chronological order.
v7fmt	Specify to display information produced by the trace facility and stored in shared memory or in a file. This trace information appears in Version 7 format.
stat	Specify to display the status of a trace facility. This status information includes the trace version, application version, number of entries, buffer size, amount of buffer used, status code, and program timestamp.
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.

Table 33. asntrc invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
-d *diag_mask*	Specifies the types of trace records to be recorded by the trace facility. Trace records are categorized by a diagnostic mask number: **1** Flow data, which includes the entry and exit points of functions. **2** Basic data, which includes all major events encountered by the trace facility. **3** Detailed data, which includes the major events with descriptions. **4** Performance data. **Important:** The higher diagnostic mask numbers are *not* inclusive of the lower diagnostic mask numbers. You can enter one or more of these numbers to construct a diagnostic mask that includes only the trace records that you need. For example, specify **-d 4** to record only performance data; specify **-d 1,4** to record only flow and performance data; specify **-d 1,2,3,4** (the default) to record all trace records. Separate the numbers with commas. Enter a diagnostic mask number of 0 (zero) to specify that no global trace records are to be recorded by the trace facility. Type **-d 0** to reset the diagnostic level before specifying new diagnostic mask numbers for a tracing facility.
-df *function_name*|*component_name diag_mask*	Specifies that a particular function or component identifier is to be traced. Type the diagnostic mask number (1,2,3,4) after the function or component identifier name. You can enter one or more of these numbers. Separate the numbers with commas.

Examples for asntrc

The following examples illustrate how to use the **asntrc** command. These examples can be run on Linux, UNIX, Windows, or z/OS operating systems.

Example 1

To trace a running Capture program:

1. Start the trace facility, specifying a trace file name with a maximum buffer and file size:
```
asntrc on -db mydb -cap -schema myschema -b 256k -fn myfile.trc -fs 500m
```
2. Start the Capture program, and let it run for an appropriate length of time.
3. While the trace facility is on, display the data directly from shared memory.

   To display the summary process and thread information from the trace facility:
```
asntrc flw -db mydb -cap -schema myschema
```

To view the flow, basic, detailed, and performance data records only from the Capture log reader:

```
asntrc fmt -db mydb -cap -schema myschema -d 0
  -df "Capture Log Read" 1,2,3,4
```

4. Stop the trace facility:

   ```
   asntrc off -db mydb -cap -schema myschema
   ```

   The trace file contains all of the Capture program trace data that was generated from the start of the Capture program until the trace facility was turned off.
5. After you stop the trace facility, format the data from the generated binary file:

   ```
   asntrc flw -fn myfile.trc
   ```

   and

   ```
   asntrc fmt -fn myfile.trc -d 0 -df "Capture Log Read" 1,2,3,4
   ```

Example 2

To start a trace facility of a Replication Alert Monitor program:

```
asntrc on -db mydb -mon -qualifier monq
```

Example 3

To trace only performance data of an Apply program:

```
asntrc on -db mydb -app -qualifier aq1 -b 256k -fn myfile.trc -d 4
```

Example 4

To trace all flow and performance data of a Capture program:

```
asntrc on dbserv1 -cap -schema myschema -b 256k
  -fn myfile.trc -d 1,4
```

Example 5

To trace all global performance data and the specific Capture log reader flow data of a Capture program:

```
asntrc on -db mydb -cap -schema myschema -b 256k -fn myfile.trc -d 4
  -df "Capture Log Read" 1
```

Example 6

To trace a running Capture program and then display and save a point-in-time image of the trace facility:

1. Start the trace command, specifying a buffer size large enough to hold the latest records:

   ```
   asntrc on -db mydb -cap -schema myschema -b 4m
   ```
2. Start the Capture program, and let it run for an appropriate length of time.
3. View the detailed point-in-time trace information that is stored in shared memory:

   ```
   asntrc fmt -db mydb -cap -schema myschema
   ```
4. Save the point-in-time trace information to a file:

   ```
   asntrc dmp myfile.trc -db mydb -cap -schema myschema
   ```
5. Stop the trace facility:

   ```
   asntrc off -db mydb -cap -schema myschema
   ```

Examples for asntrc with 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, and 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 34 defines the invocation parameters for the **asntrep** command.

Table 34. asntrep invocation parameter definitions for Linux, UNIX, and Windows operating systems

Parameter	Definition
DB=*server*	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: **Q Replication** The value is the name of the Q Capture server, which contains the IBMQREP_SUBS table. **SQL Replication** The value is the name of the Apply control server, which contains the IBMSNAP_SUBS_MEMBR table. z/OS The value of this parameter is a location name.
DB2_SUBSYSTEM=*subsystem*	z/OS Specifies the name of the subsystem where you run the asntrep utility.
SCHEMA=*schema*	Specifies the schema of the Q Capture control tables for Q Replication, or the Apply control tables for SQL Replication.
DIFF_SCHEMA=*difference_table_schema*	Specifies the schema that qualifies the difference table. The default is ASN.
DIFF_TABLESPACE=*tablespace*	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 **asntrep** command was run.

Table 34. asntrep invocation parameter definitions for Linux, UNIX, and Windows operating systems (continued)

Parameter	Definition
WHERE=*WHERE_clause*	Specifies a SQL WHERE clause that uniquely identifies one row of the control table that stores information about the source and target tables that you are synchronizing. The WHERE clause must be in double quotation marks. The value of this parameter differs depending on whether you are using Q Replication or SQL Replication: **Q Replication** The WHERE clause specifies a row in the IBMQREP_SUBS table and uses the SUBNAME column to identify the Q subscription that contains the source and target tables. **SQL Replication** The WHERE clause specifies a row in the IBMSNAP_SUBS_MEMBR table and uses the SET_NAME, APPLY_QUAL, TARGET_SCHEMA, and TARGET_TABLE columns to identify the subscription set member that contains the source and target tables.
DIFF_PATH=*log_path*	Specifies the location where you want the asntrep 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.
PWDFILE=*filename*	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 asnpwd.aut (the name of the password file that is created by the **asnpwd** command). The asntrep 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.
DIFF=*table_name*	Specifies the name of the table that was created in the source database by the **asntdiff** command to store differences between the source and target tables. The information that is stored in this table is used to synchronize the source and target tables.

Examples for asntrep

The following examples illustrate how to use the **asntrep** command.

Example 1

In Q Replication, to synchronize a source and target table that are specified in a Q subscription named my_qsub, on a Q Capture server named source_db, with a Q Capture schema of asn, and whose differences are stored in a table called q_diff_table:

```
asntrep db=source_db schema=asn where="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 22. System commands for SQL replication (System i)

Some replication commands are specific to the System i operating system on System i 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 (System i)

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

```
►►──ADDDPRREG──SRCTBL──(──library-name/file-name──)──────────────────────────►

►──┬────────────────────────────────────────┬──┬──────────────────────────────────────┬──►
   │                ┌─ASN──────────┐        │  │           ┌─*SRCTBL──────┐           │
   └─CAPCTLLIB──(───┴─library-name─┴───)────┘  └─CDLIB──(───┴─library-name─┴───)───────┘

►──┬─────────────────────────────┬──┬───────────────────────────────────┬──►
   │           ┌─*DEFAULT─┐      │  │            ┌─*USERTABLE────┐      │
   └─CDNAME──(─┴─cdname───┴──)───┘  └─SRCTYPE──(─┼─*POINTINTIME──┼──)───┘
                                                 ├─*BASEAGR──────┤
                                                 ├─*CHANGEAGR────┤
                                                 ├─*REPLICA──────┤
                                                 ├─*USERCOPY─────┤
                                                 └─*CCD──────────┘

►──┬────────────────────────────┬──┬────────────────────────────────────┬──►
   │            ┌─*YES─┐        │  │          ┌─*NONE───────────────┐   │
   └─REFRESH──(─┴─*NO──┴───)────┘  └─TEXT──(──┴─'──description──'───┴──)┘

►──┬────────────────────────────────────────────┬──┬────────────────────────────┬──►
   │           ┌─*ALL────────────────────┐      │  │           ┌─*NO──┐         │
   │           ├─*NONE───────────────────┤      │  └─CAPRRN──(─┴─*YES─┴───)─────┘
   │           │ ┌──────────────────┐    │      │
   │           │ ▼             (1)  │    │      │
   └─CAPCOL──(─┴───column-name──────┴────┴──)───┘
```

```
►─┬─────────────────────────────┬─┬──────────────────────────────────────┬──────────────────►
  │           ┌─*AFTER─┐        │ │                 ┌─*DEFAULT──┐        │
  └─IMAGE──(──┴─*BOTH──┴──)─────┘ └─PREFIX──(──┼─*NULL─────┼──)──────────┘
                                                    └─character─┘

►─┬──────────────────────────────────────────┬─┬─────────────────────────────────┬──────────►
  │                                           │ │                 ┌─*YES─┐          │
  │               ┌─*YES───────┐              │ └─COMPLETE──(──┴─*NO──┴──)───────┘
  └─CONDENSED──(──┼─*NO────────┼──)───────────┘
                  └─*AGGREGATE─┘

►─┬───────────────────────────────────┬─────────────────────────────────────────────────────►
  │               ┌─*LOCAL──┐          │
  └─SRCTBLRDB──(──┴─rdbname─┴──)──────┘

►─┬─────────────────────────────────────────────────────┬───────────────────────────────────►
  │            ┌─*SRCTBL──────────────────────┐          │
  └─RMTJRN──(──┴─library-name/journal-name────┴──)──────┘

►─┬──────────────────────────────────────┬─┬─────────────────────────────────┬──────────────►
  │                                       │ │                  ┌─*NO──┐         │
  │              ┌─*NONE──────┐           │ └─UPDDELINS──(──┴─*YES─┴──)──────┘
  └─CONFLICT──(──┼─*STANDARD──┼──)────────┘
                 └─*ENHANCED──┘

►─┬──────────────────────────────────────┬─┬─────────────────────────────┬──────────────────►
  │              ┌─*ALLCHG────┐           │ │              ┌─*YES─┐        │
  └─GENCDROW──(──┴─*REGCOLCHG─┴──)────────┘ └─RECAP──(──┴─*NO──┴──)──────┘

►─┬───────────────────────────────┬──────────────────────────────────────────────────────►◄
  │               ┌─*NO──┐         │
  └─STOPONERR──(──┴─*YES─┴──)─────┘
```

Notes:

1 You can specify up to 300 column names.

Table 35 lists the invocation parameters.

Table 35. ADDDPRREG command parameter definitions for System i

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 a System i library or collection that is externally defined and in single format. This parameter is required. *library-name/file-name* Represents the qualified name of the table that you want to register.
CAPCTLLIB	Specifies the Capture schema, which is the name of the library in which the Capture control tables reside. **ASN (default)** The Capture control tables reside in the ASN library. *library-name* The name of the library that contains the Capture control tables. You can create this library by using the **CRTDPRTBL** command with the **CAPCTLLIB** parameter.

Table 35. ADDDPRREG command parameter definitions for System i (continued)

Parameter	Definition and prompts
CDLIB	Specifies the library in which the change-data (CD) table for this registered source is created. ***SRCTBL (default)** Creates the CD table in the library in which the source table resides. *library-name* Creates the CD table in this specified library name.
CDNAME	Specifies the name of the change-data (CD) table. ***DEFAULT (default)** Creates the CD table with the default name, which is based on the current timestamp. For example, if the current timestamp is January 23, 2002 at 09:58:26, the default name is ASN020123095826CD. *cdname* Creates the CD table with this specified name.

Table 35. ADDDPRREG command parameter definitions for System i (continued)

Parameter	Definition and prompts
SRCTYPE	Specifies the type of source table that you are registering. Choose a source type based on your replication configuration: • Use the default of USERTABLE for a basic data distribution or a data consolidation configuration. • Use REPLICA for an update-anywhere configuration. • Use POINTINTIME, BASEAGR, CHANGEAGR, USERCOPY, or CCD if you have a multi-tier configuration and want the target table to be a source for a subsequent tier in your replication configuration. If you are registering an existing target table as a source, the registration fails if the target table does not contain the IBMSNAP table columns indicated by the specified source type. ***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 System i column identifier of either IBMSNAP or IBMQSQ. ***POINTINTIME** A point-in-time copy table, which includes content that matches all or part of the content of a source table and a DB2 DataPropagator for System i system column that identifies the time when a particular row was last inserted or updated at the source system. The table must contain the IBMSNAP_LOGMARKER timestamp column and can optionally contain an INTEGER column called IBMQSQ_RRN. ***BASEAGR** A base aggregate copy, which contains data aggregated at intervals from a user table or from a point-in-time table. The base aggregate table must contain the IBMSNAP_HLOGMARKER and IBMSNAP_LLOGMARKER timestamp columns. ***CHANGEAGR** A change aggregate copy table, which contains data aggregations that are based on changes recorded for a source table. The table must contain the IBMSNAP_HLOGMARKER and IBMSNAP_LLOGMARKER timestamp columns. ***REPLICA** A target table for a replica subscription. Register this type of table so that changes from the target table are replicated back to the original source table. This table cannot contain any DB2 DataPropagator for System i system columns or any columns that start with the DB2 DataPropagator for System i column identifier of either IBMSNAP or IBMQSQ. The table contains all of the columns from the original source table. ***USERCOPY** A target table with content that matches all or part of the content of a source table. The user copy table contains only user data columns.

Table 35. ADDDPRREG command parameter definitions for System i (continued)

Parameter	Definition and prompts
SRCTYPE (Continued)	***CCD** A consistent-change data (CCD) table, which contains transaction-consistent data from the source table. The table must contain columns that are defined as follows: • IBMSNAP_INTENTSEQ CHAR(10) FOR BIT DATA NOT NULL • IBMSNAP_OPERATION CHAR(1) NOT NULL • IBMSNAP_COMMITSEQ CHAR(10) FOR BIT DATA NOT NULL • IBMSNAP_LOGMARKER TIMESTAMP NOT NULL
REFRESH	Specifies whether the full-refresh capability is enabled. You can use this value to turn off the capability of the Apply program to perform a full refresh from the source database. ***YES (default)** Full refreshes are allowed. ***NO** Full refreshes are not allowed. If the target table is a base aggregate or change aggregate, you should set this parameter to *No.
TEXT	Specifies the textual description that is associated with this registration. ***NONE (default)** No description is associated with the entry. *description* The textual description of this registration. You can enter a maximum of 50 characters and must enclose the text in single quotation marks.
CAPCOL	Specifies the columns for which changes are captured for this registered table. ***ALL (default)** Changes are captured for all columns. ***NONE** Changes are not captured for this table. Use this value to specify that you want this table registered for full refresh only. The change-data (CD) table is not created with this registered table, and the Capture program will not capture changes for the table. *column-name* The column names for which changes are captured. You can type up to 300 column names. Separate the column names with spaces.
CAPRRN	Specifies whether the relative record number (RRN) of each changed record is captured. ***NO (default)** The relative record number is not captured. ***YES** The relative record number is captured. An additional column called IBMQSQ_RRN is created in the change-data (CD) table. Set this parameter to *YES only if there are no unique keys in the source table.

Table 35. ADDDPRREG command parameter definitions for System i (continued)

Parameter	Definition and prompts
IMAGE	Specifies whether the change-data (CD) table contains both before and after images of the changes to the source table. This applies globally to all columns specified on the Capture columns (**CAPCOL**) parameter. This **IMAGE** parameter is not valid when the **CAPCOL** parameter is set to *NONE. The source table must be journaled with *BOTH images even if you specify *AFTER on this parameter. ***AFTER (default)** The Capture program records only after images of the source table in the CD table. ***BOTH** The Capture program records both before images and after images of the source table in the CD table.
PREFIX	Specifies the prefix character identifying before-image column names in the change-data (CD) table. You must ensure that none of the registered column names of the source table begins with this prefix character. ***DEFAULT (default)** The default prefix (@) is used. ***NULL** No before images are captured. This value is not valid if the **IMAGE** parameter is set to *BOTH. *character* Any single alphabetic character that is valid in an object name.
CONDENSED	Specifies whether the source table is condensed. A condensed table contains current data with no more than one row for each primary key value in the table. ***YES (default)** The source table is condensed. ***NO** The source table is not condensed. ***AGGREGATE** The source table type is either *BASEAGR (base aggregate) or *CHANGEAGR (change aggregate). If this value is used, you must set the **COMPLETE** parameter to *No
COMPLETE	Specifies whether the source table is complete, which means that the table contains a row for every primary key value of interest. ***YES (default)** The source table is complete. ***NO** The source table is not complete.

Table 35. ADDDPRREG command parameter definitions for System i (continued)

Parameter	Definition and prompts
SRCTBLRDB	Specifies whether you want to use remote journaling, in which the source table and the remote journal reside on different systems. Use this parameter to specify the location of the source table. ***LOCAL (default)** The source table resides locally (on the machine where you are running the **ADDDPRREG** command). *rdbname* The name of the relational database where the source table exists. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this relational database name.
RMTJRN	Specifies the name of the remote journal when the name of this journal and the name of the journal on the source system are different. You must issue this command from the system where the remote journal resides. ***SRCTBL (default)** The remote journal name is the same as the journal name of the source table. *library-name/journal-name* The qualified library and journal name that reside on this system and are used for journaling the remote source table. You can specify a remote journal name only if you specified a remote source table location by using the **SRCTBLRDB** parameter.
CONFLICT	Specifies the conflict level that is used by the Apply program when detecting conflicts in a replica subscription. ***NONE (default)** No conflict detection. ***STANDARD** Moderate conflict detection. The Apply program searches for conflicts in rows that are already captured in the replica change-data (CD) tables. ***ENHANCED** Enhanced conflict detection. This option provides the best data integrity among all replicas and source tables.
UPDDELINS	Determines how the Capture program stores updated source data in the change-data (CD) table. ***NO (default)** The Capture program stores each source change in a single row in the CD table. ***YES** The Capture program stores each source change by using two rows in the CD table, one for the delete and one for the insert. The Apply program processes the delete row first and the insert row second.

Table 35. ADDDPRREG command parameter definitions for System i (continued)

Parameter	Definition and prompts
GENCDROW	Specifies whether the Capture program captures changes from all rows in the source table. ***ALLCHG (default)** The Capture program captures changes from all rows in the source table (including changes in unregistered columns) and adds these changes to the change-data (CD) table. ***REGCOLCHG** The Capture program captures changes only if the changes occur in registered columns. The Capture program then adds these rows to the CD table. You cannot specify *REGCOLCHG if the **CAPCOL** parameter is set to *ALL or *NONE.
RECAP	Specifies whether the changes made by the Apply program are recaptured by the Capture program. ***YES (default)** Changes made to the source table by the Apply program are captured and entered into the change-data (CD) table. ***NO** Changes that were made to the source table by the Apply program are not captured and, therefore, do not appear in the CD table. You should use this option when registering REPLICA type tables.
STOPONERR	Specifies whether the Capture program stops when it encounters an error.[1] ***NO (default)** The Capture program does not stop when it encounter an error. The Capture program issues messages, deactivates the registration that caused the error, and then continues processing. ***YES** The Capture program issues messages and then stops when it encounters an error.

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 200A*xxxx* where *xxxx* is the reason code. To set the registration back to the Action ('A') state, perform the following steps:

   - Correct the ASN200A message. Refer to the appropriate System i documentation for the corrected action.
   - Use the Replication Center or the System i 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 (System i)

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

```
►►—ADDDPRSUB—APYQUAL—(—apply-qualifier—)—SETNAME—(—set-name—)——————————————►
```

```
              ┌─*NONE──────────────────┐                            ┌─*NONE──────────────────┐
►─SRCTBL──(───┴─library-name/file-name─┴──)──TGTTBL──(───┴─library-name/file-name─┴──)────────────────►

►──┬─────────────────────────────────────┬─┬─────────────────────────────────────┬─────────────────────►
   │                ┌─*LOCAL───┐         │ │                ┌─*LOCAL───┐         │
   └─CTLSVR──(──────┴─rdb-name─┴───)─────┘ └─SRCSVR──(──────┴─rdb-name─┴───)─────┘

►──┬───────────────────────────────────────┬─┬─────────────────────────────────────┬──────────────────►
   │                 ┌─*USERCOPY────┐      │ │                ┌─*INTERVAL─┐        │
   └─TGTTYPE──(──────┼─*POINTINTIME─┼──)───┘ └─TIMING──(──────┼─*EVENT────┼───)───┘
                     ├─*BASEAGR─────┤                         └─*BOTH─────┘
                     ├─*CHANGEAGR───┤
                     ├─*CCD─────────┤
                     └─*REPLICA─────┘

►──┬───────────────────────────────────┬─────────────────────────────────────────────────────────────►
   │              ┌─*NONE──────┐       │
   └─EVENT──(─────┴─event-name─┴──)────┘

►──┬───────────────────────────────────────────────────────────────────────────────┬─────────────────►
   └─INTERVAL──(──num *MIN) (num *HOUR) (num *DAY) (num *WEEK──)────────────────────┘

►──┬────────────────────────────────┬─┬───────────────────────────────────┬─┬─────────────────────────┬─►
   │                ┌─*YES─┐        │ │                ┌─*YES─┐           │ │            ┌─*YES─┐    │
   └─ACTIVATE──(────┴─*NO──┴───)────┘ └─CRTTGTTBL──(────┴─*NO──┴───)──────┘ └─CHKFMT──(───┴─*NO──┴──)─┘

►──┬────────────────────────────────────────┬─┬──────────────────────────────────────────┬───────────►
   │                 ┌─ASN──────────┐       │ │               ┌─*CAPCTLLIB───┐           │
   └─CAPCTLLIB──(────┴─library-name─┴───)───┘ └─TGTCCLIB──(───┴─library-name─┴───)───────┘

►──┬────────────────────────────────────┬─┬─────────────────────────────────────────────┬───────────►
   │              ┌─*NONE───────┐       │ │                                             │
   └─FEDSVR──(────┴─server-name─┴──)────┘ │                ┌─*DEFAULT─────────┐         │
                                          └─CMTCNT──(──────┼─*NULL────────────┼───)─────┘
                                                           └─num-transactions─┘

►──┬──────────────────────────────────┬─┬─────────────────────────────────────────────────┬─────────►
   │                 ┌─*NO──┐         │ │                ┌─*ALL───────────────────┐       │
   └─TGTKEYCHG──(────┴─*YES─┴───)─────┘ └─COLUMN──(──────┼─*NONE──────────────────┼───)───┘
                                                         │ ┌───────────────────┐  │
                                                         │ ▼               (1)   │  │
                                                         └───column-name───────┴──┘

►──┬──────────────────────────────┬─┬─────────────────────────────────────────────┬─────────────────►
   │             ┌─*YES─┐         │ │               ┌─*SRCTBL────────────────┐    │
   └─UNIQUE──(───┴─*NO──┴───)─────┘ └─KEYCOL──(─────┼─*RRN───────────────────┼──)─┘
                                                    ├─*NONE──────────────────┤
                                                    │ ┌───────────────────┐  │
                                                    │ ▼               (2)   │  │
                                                    └───column-name───────┴──┘

►──┬──────────────────────────────────────────────────────────────┬─────────────────────────────────►
   │               ┌─*COLUMN───────────────────────────────┐      │
   │               │ ┌─────────────────────────────────┐   │      │
   │               │ ▼                            (3)  │   │      │
   └─TGTCOL──(─────┴───(column-name new-name)──────────┴───┴──)───┘

►──┬───────────────────────────────────────────────────────────────────┬─┬──────────────────────────┬─►
   │                 ┌─*NONE──────────────────────────────────┐        │ │               ┌─*NO──┐   │
   │                 │ ┌──────────────────────────────────┐   │        │ └─ADDREG──(─────┴─*YES─┴─)─┘
   │                 │ ▼                            (4)   │   │        │
   └─CALCCOL──(──────┴───(column-name expression)─────────┴───┴──)─────┘
```

```
►─┬──────────────────────────────────┬──────────────────────────────────────────────────────────►
  │              ┌─*ALL─────────┐    │
  └─ROWSLT──(────┴─WHERE-clause─┴──)─┘

►─┬────────────────────────────────────────────────────────────────┬────────────────────────────►
  │           ┌─0────────────────────────────────────────────────┐ │
  └─MAXSYNCH──┴─(num *MIN) (num *HOUR) (num *DAY) (num *WEEK)────┴─┘

►─┬────────────────────────────────────────────────────────────────────────────────────────┬─────►
  │              ┌─*NONE────────────────────────┐              ┌─*NONE──────────────────┐    │
  │              │ ┌──────────────────────────┐ │  ┌─*TGTSVR─┐ │ ┌────────────────────┐ │    │
  │              │ ▼                      (5) │ │  │         │ │ ▼                (6) │ │    │
  └─SQLBEFORE──(─┴───SQL-statement────────────┴─┴──┴─*SRCSVR─┴─┴───SQL-states─────────┴─┴──)─┘

►─┬──────────────────────────────────────────────────────────────────────────────────────┬──────►◄
  │             ┌─*NONE────────────────────────┐              ┌─*NONE──────────────────┐   │
  │             │ ┌──────────────────────────┐ │              │ ┌────────────────────┐ │   │
  │             │ ▼                      (7) │ │  ┌─*TGTSVR─┐ │ ▼                (8) │ │   │
  └─SQLAFTER──(─┴───SQL-statement────────────┴─┴──┴─────────┴─┴───SQL-states─────────┴─┴─)─┘
```

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 36 lists the invocation parameters.

Table 36. ADDDPRSUB command parameter definitions for System i

Parameter	Definition and prompts
APYQUAL	Specifies the Apply qualifier that identifies which Apply program processes this subscription set. Subscription sets under an Apply qualifier run in a separate job. This parameter is required. *apply-qualifier* The name of the Apply qualifier.
SETNAME	Specifies the subscription-set name. This parameter is required. *set-name* The name of the subscription set. The subscription-set name that you enter must be unique for the specified Apply qualifier or the **ADDDPRSUB** command produces an error. Because the Apply program handles the set of target tables as a group, when one target table fails for any reason, the entire subscription set fails.

Table 36. ADDDPRSUB command parameter definitions for System i (continued)

Parameter	Definition and prompts
SRCTBL	Specifies the name of the source table that is used to copy information into your subscription set. You must register this table at the Capture control server before this table can become a member of a subscription set. This parameter is required. ***NONE (default)** This subscription set does not have a source member. Use when creating a subscription set without members. *library-name/file-name* The qualified name of the source table. Use when creating a subscription set with one member.
TGTTBL	Specifies the name of the target table. The target table is automatically created if you set the **CRTTGTTBL** parameter to *YES and the target table does not already exist. This parameter is required. ***NONE (default)** This subscription set does not have a target member. Use when creating a subscription set without members. *library-name/file-name* The qualified name of the target table. Use when creating a subscription set with one member.
CTLSVR	Specifies the relational database name of the system that contains the Apply control tables. ***LOCAL (default)** The Apply control tables reside locally (on the machine from which you are running the **ADDDPRSUB** command). *rdb-name* The name of the relational database where the Apply control tables reside. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this name.
SRCSVR	Specifies the relational database name of the system that contains the Capture control tables. ***LOCAL (default)** The source table is registered on the local machine (the machine from which you are running the **ADDDPRSUB** command). *rdb-name* The name of the relational database where the Capture control tables reside. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this name.

Table 36. ADDDPRSUB command parameter definitions for System i (continued)

Parameter	Definition and prompts
`TGTTYPE`	Specifies the target table type. After you create a target table as one of these types, you can use this parameter value on the **SRCTBL** parameter of the Add DPR Registration (**ADDDPRREG**) command to register this target table as a source table for multi-tier replication. ***USERCOPY (default)** The target table is a user copy, which is a target table with content that matches all or part of the content of a source table. A user copy is handled like a point-in-time copy but does not contain any of the DB2 DataPropagator for System i system columns that are present in the point-in-time target table. This value is not valid when a value of *RRN is specified on the **KEYCOL** parameter. The table that you specified with the **SRCTBL** parameter must be one of the following types: user database, point-in-time copy, or consistent-change data (CCD). **Important:** If the target table already exists, DB2 DataPropagator for System i does not automatically journal changes to it. You must start journaling outside of DB2 DataPropagator for System i. ***POINTINTIME** The target table is a point-in-time copy. A point-in-time copy is a target table with content that matches all or part of the content of the source table and includes the DB2 DataPropagator for System i system column (IBMSNAP_LOGMARKER), which identifies when a particular row was inserted or updated at the Capture control server. ***BASEAGR** The target table is a base aggregate copy, which is a target table that contains data that is aggregated (calculated) from a source table. The source table for a base aggregate target must be either a user table or a point-in-time table. This target table must contain the IBMSNAP_HLOGMARKER and IBMSNAP_LLOGMARKER system timestamp columns. ***CHANGEAGR** The table is a change aggregate copy, which is a target table that contains data that is aggregated (calculated) based on the contents of a change-data (CD) table. This target table is created with the IBMSNAP_HLOGMARKER and IBMSNAP_LLOGMARKER system timestamp columns.

Table 36. ADDDPRSUB command parameter definitions for System i (continued)

Parameter	Definition and prompts
TGTTYPE (Continued)	***CCD** The table is a consistent-change data (CCD) table, which is a target table created from a join of data in the change-data (CD) table and the unit-of-work (UOW) table. A CCD table provides transaction-consistent data for the Apply program and must include the following columns: • IBMSNAP_INTENTSEQ • IBMSNAP_OPERATION • IBMSNAP_COMMITSEQ • IBMSNAP_LOGMARKER ***REPLICA** The target table is a replica table, which is used only for update-anywhere replication. The replica target table receives changes from the master source table, and changes to the replica target table are propagated back to the master source table. A replica table is automatically registered as a source table.
TIMING	Specifies the type of timing (scheduling) that the Apply program uses to process the subscription set. ***INTERVAL (default)** The Apply program processes the subscription set at a specific time interval (for example, once a day). ***EVENT** The Apply program processes the subscription set when a specific event occurs. ***BOTH** The Apply program processes the subscription set either at a specific interval or when an event occurs, whichever occurs first.
EVENT	Specifies an event. The event that you enter must match an event name in the IBMSNAP_SUBS_EVENT) table. ***NONE (default)** No event is used. *event-name* A unique character string that represents an event described in the IBMSNAP_SUBS_EVENT table.

Table 36. ADDDPRSUB command parameter definitions for System i (continued)

Parameter	Definition and prompts
INTERVAL	Specifies the time interval (weeks, days, hours, and minutes) from start time to start time between refreshes of the target copy. This is a two-part value. The first part is a number; the second part is the unit of time: ***MIN** Minutes ***HOUR** Hours ***DAY** Days ***WEEK** Weeks You can specify combinations of numbers with units of time. For example, ((2 *WEEK) (3 *DAY) (35 *MIN)) specifies a time interval of two weeks, three days, and 35 minutes. If you specify multiple occurrences of the same unit of time, the last occurrence is used.
ACTIVATE	Specifies whether the subscription set is active. The Apply program does not process this subscription set unless this parameter is set to *YES. ***YES (default)** The subscription set is active. ***NO** The subscription set is not active.
CRTTGTTBL	Specifies whether the target table (or view) is created. ***YES (default)** Creates the target table (or view) if it does not exist. Otherwise, the existing table or view becomes the target, and the format of this existing table or view is checked if the value of the **CHKFMT** parameter is set to *YES. An additional index, with the values that you specified by the **UNIQUE** and **KEYCOL** parameters, is created for a target table if no such index currently exists. The command fails if an existing target table contains rows that violate the conditions of the additional index. ***NO** Does not create the target table or view. You must create the table or view with the correct attributes before starting the Apply program. If the table or view exists and you set **CHKFMT** to *YES, the **ADDDPRSUB** command ensures that the format of the existing table matches the subscription-set definition that you set. If **CHKFMT** is *NO, you must ensure that the format of the existing table matches the subscription-set definition. **Important:** If the table or view already exists, DB2 DataPropagator for System i does not automatically journal changes to the existing object. You must start journaling outside of DB2 DataPropagator for System i.

Table 36. ADDDPRSUB command parameter definitions for System i (continued)

Parameter	Definition and prompts
CHKFMT	Specifies whether DB2 DataPropagator for System i checks the subscription set and the target table to ensure that the columns match. This parameter is ignored if the **CRTTGTTBL** parameter is *YES; this parameter is also ignored if the **CRTTGTTBL** parameter is set to *NO and the target table does not exist. ***YES (default)** DB2 DataPropagator for System i verifies that the columns defined for this subscription set match the columns in the target table. This command fails if a mismatch is detected. ***NO** DB2 DataPropagator for System i ignores the differences between the subscription set and the existing target table. You must ensure that the target table is compatible with the subscription set.
CAPCTLLIB	Specifies the Capture schema, which is the name of the library in which the Capture control tables reside. These Capture control tables process the source for this subscription set. **ASN (default)** The Capture control tables reside in the ASN library. *library-name* The name of a library that contains the Capture control tables. This is the library in which the source table was registered.
TGTCCLIB	Specifies the target control library. ***CAPCTLLIB (default)** The target control library is the same library in which the Capture control tables reside. *library-name* The name of a library that contains the target control tables. If you are using a target table as a source for another subscription set (such as an external CCD table), this parameter value is the Capture schema when this table is used as a source.
FEDSVR	Specifies whether a federated database system is the source for this subscription set. ***NONE (default)** The source server is not a federated database system. *server-name* The name of the federated database system for this subscription set (for non-DB2 relational sources).

Table 36. ADDDPRSUB command parameter definitions for System i (continued)

Parameter	Definition and prompts
`CMTCNT`	Specifies the commitment count, which is the number of transactions that the Apply program processes before a commit. **`*DEFAULT (default)`** The command determines the value to use. If the **TGTTYPE** is set to *REPLICA, then the **CMTCNT** is zero (0). If the **TGTTYPE** is anything other than *REPLICA, the **CMTCNT** is null. **`*NULL`** The subscription set is read-only. The Apply program will fetch answer sets for the subscription-set members one member at a time, until all data has been processed and then will issue a single commit for the entire subscription set. *`num-transactions`* Specifies the number of transactions processed before the Apply program commits the changes. This parameter is valid only if the **TGTTYPE** parameter is set to *REPLICA.
`TGTKEYCHG`	Specifies how the Apply program handles updates when changes occur in source columns that are part of the target key columns for the target table. This parameter works in conjunction with the **USEDELINS** parameter on the **`ADDDPRREG`** command: • If **USEDELINS** is YES and **TGTKEYCHG** is YES, updates are not allowed. • If **USEDELINS** is YES and **TGTKEYCHG** is NO, updates become delete and insert pairs. • If **USEDELINS** is NO and **TGTKEYCHG** is YES, the Apply program handles this condition with special logic. • If **USEDELINS** is NO and **TGTKEYCHG** is NO, the Apply program processes the changes as normal updates. **`*NO (default)`** Updates to the source table are staged by the Capture program and processed by the Apply program to the target table. **`*YES`** The Apply program updates the target table based on the before images of the target key column, meaning that the Apply program changes the predicate to the old values instead of the new.

Table 36. ADDDPRSUB command parameter definitions for System i (continued)

Parameter	Definition and prompts
COLUMN	Specifies the columns to be included in the target table. The column names must be unqualified. Choose the column names from the list of column names that you specified with the **CAPCOL** parameter when you registered the source table. If you set the **IMAGE** parameter to *BOTH when registering this table, you can specify before-image column names. The before-image column names are the original column names with a prefix. This prefix is the character that you specified in the **PREFIX** parameter of the **ADDDPRREG** command. **`*ALL (default)`** All of the columns that you registered in the source are included in the target table. **`*NONE`** No columns from the source table are included in the target table. You can use *NONE when you want only computed columns in the target table. This value is required if the **CALCCOL** parameter contains summary functions but no GROUP BY is performed. *`column-name`* The names of up to 300 source columns that you want to include in the target table. Separate the column names with spaces.
UNIQUE	Specifies whether the target table has unique keys as indicated by the **KEYCOL** parameter. **`*YES (default)`** The target table supports one net change per key; only one row exists in the target table for that key regardless of how many changes are made to the key. This value specifies that the table contains current data rather than a history of changes to the data. A condensed table includes no more than one row for each primary key value in the table and can be used to supply current information for a refresh. **`*NO`** The target table supports multiple changes per key. The changes are appended to the target table. This value specifies that the table contains a history of changes to the data rather than current data. A non-condensed table includes more than one row for each key value in the table and can be used to supply a history of changes to the data. A non-condensed table cannot supply current data for a refresh.

Table 36. ADDDPRSUB command parameter definitions for System i (continued)

Parameter	Definition and prompts
KEYCOL	Specifies columns that describe the key of the target table. The column names must be unqualified. For *POINTINTIME, *REPLICA, and *USERCOPY target tables (as specified on the **TGTTYPE** parameter), you must identify one or more columns as the target key for the target table. This target key is used by the Apply program to identify each unique row that changes during change-capture replication. ***SRCTBL (default)** The key columns in the target table are the same as those in the source table. The **ADDDPRREG** command uses the key that is specified in the source table if the source table is keyed. The following key columns are used: • Key columns that you defined through DDS when creating the table with the Create Physical File (**CRTPF**) command • Primary and unique keys that you defined with the CREATE TABLE and ALTER TABLE SQL statements • Unique keys that you defined with the CREATE INDEX SQL statements If you use a column as a key more than once and with different ordering, the target table key is defined with ascending order. ***RRN** The key column in the target table is the IBMQSQ_RRN column. The target table is created with an IBMQSQ_RRN column, and this column is used as the key. When the Apply program runs, if the source table is a user table and the target table is a point-in-time or user copy, the IBMQSQ_RRN column in the target table is updated with the relative record number of the associated record in the source table. Otherwise, the IBMQSQ_RRN column in the target table is updated with the value of the IBMQSQ_RRN column in the source table. ***NONE** The target copy does not contain a target key. You cannot specify *NONE if the target table type is *POINTINTIME, *REPLICA, or *USERCOPY. *column-name* The names of the target columns that you want to use as the target key columns. You can specify up to 120 column names. Separate the column names with spaces.

Table 36. ADDDPRSUB command parameter definitions for System i (continued)

Parameter	Definition and prompts
TGTCOL	Specifies the new names for all the columns that the Apply program updates in the target table. These names override the column names taken from the source table. The column names must be unqualified. If you specified a value of *NONE for the **COLUMN** parameter, do not use this parameter. Use this parameter to give more meaningful names to the target table columns. Specify each source column name and the name of the corresponding column on the target table. ***COLUMN (default)** The target columns are the same as the columns you specified in the **COLUMN** parameter. *column-name* The column names from the source table that you want to change at the target. You can list up to 300 column names. *new-name* The new names of the target columns. You can list up to 300 new column names. If you do not use this parameter, the name of the column on the target table will be the same as the source column name.
CALCCOL	Specifies the list of user-defined or calculated columns in the target table. The column names must be unqualified. Enclose each column name and expression pair in parenthesis. You must specify a column name for each SQL expression. If you want to define any column as an SQL expression without a GROUP BY statement, you must set the **COLUMN** parameter to *NONE. ***NONE (default)** No user-defined or calculated columns are included in the target table. *column-name* The column names of the user-defined or calculated columns in the target table. You can list up to 100 column names. *expression* The expressions for the user-defined or calculated columns in the target table. You can list up to 100 SQL column expressions.
ADDREG	Specifies whether the target table is automatically registered as a source table. Use this parameter to register CCD target type tables. ***NO (default)** The target table is not registered as a source table. DB2 DataPropagator for System i ignores this parameter value if the target type is *REPLICA. Replica target tables are always automatically registered as source tables. ***YES** The target table is registered as a source table. This command fails if you already registered the target table. Do not set this parameter to *YES if the target table type is *USERCOPY, *POINTINTIME, *BASEAGR, or *CHANGEAGR. If you set the **CRTTGTTBL** parameter to *NO, you must create the target table before attempting to register it as a source.

Table 36. ADDDPRSUB command parameter definitions for System i (continued)

Parameter	Definition and prompts
ROWSLT	Specifies the predicates to be placed in an SQL WHERE clause. The Apply program uses these predicates to determine which rows in the change-data (CD) table of the source to apply to the target table. Use this parameter if you want only a subset of the source changes to be replicated to the target table. ***ALL (default)** The Apply program applies all changes in the CD table to the target table. *WHERE-clause* The SQL WHERE clause that specifies which rows from the CD table the Apply program applies to the target table. Do not include the WHERE keyword; it is implied on this parameter. This WHERE clause must be valid on the data server you are using to run the clause. **Note:** The WHERE clause on this parameter is unrelated to any WHERE clauses specified on the SQLBEFORE or SQLAFTER parameters.
MAXSYNCH	Specifies the maximum synch minutes. This parameter is the time-threshold limit used to regulate the amount of change data that the Capture and Apply programs process during a subscription cycle. You can specify the time-threshold limit by using a two-part value. The first part is a number; the second part is the unit of time: ***MIN** Minutes ***HOUR** Hours ***DAY** Days ***WEEK** Weeks You can specify combinations of numbers with units of time. For example, ((1 *WEEK) (2 *DAY) (35 *MIN)) specifies a time interval of one week, two days, and 35 minutes. If you specify multiple occurrences of the same unit of time, the last occurrence is used. The default is zero (0), which indicates that all of the change data is to be applied.

Table 36. ADDDPRSUB command parameter definitions for System i (continued)

Parameter	Definition and prompts
SQLBEFORE	Specifies the SQL statements that run before the Apply program refreshes the target table. This parameter has three elements: Element 1: SQL code ***NONE (default)** No SQL statement is specified. *SQL-statement* The SQL statement that you want to run. Ensure that the syntax of the SQL statement is correct. DB2 DataPropagator for System i does not validate the syntax. In addition, you must use the proper SQL naming conventions. SQL file references must be in the form of LIBRARY.FILE instead of the system naming convention (LIBRARY/FILE). You can specify up to three SQL statements. Element 2: Server to run on ***TGTSVR (default)** The SQL statement runs at the target server on which the target table is located. ***SRCSVR** The SQL statement runs at the Capture control server on which the source table is located. Element 3: Allowed SQLSTATE values ***NONE (default)** Only an SQLSTATE value of 00000 is considered successful. *SQL-states* A list of one to ten allowable SQLSTATE values. Separate the SQLSTATE values with spaces. An SQLSTATE value is a five-digit hexadecimal number ranging from 00000 to FFFFF. The SQL statement is successful if it completes with an SQLSTATE value of 00000 or with one of the allowable SQLSTATE values that you listed.

Table 36. ADDDPRSUB command parameter definitions for System i (continued)

Parameter	Definition and prompts
SQLAFTER	Specifies SQL statements that run after the Apply program refreshes the target table. This parameter has three elements: Element 1: SQL code ***NONE (default)** No SQL statement is specified. *SQL-statement* The SQL statement that you want to run. Ensure that the syntax of the SQL statement is correct. DB2 DataPropagator for System i does not validate the syntax. In addition, you must use the proper SQL naming conventions. SQL file references must be in the form of LIBRARY.FILE instead of the system naming convention (LIBRARY/FILE). You can specify up to three SQL statements. Element 2: Server to run on ***TGTSVR (default)** The SQL statement runs at the target server on which the target table is located. Element 3: Allowed SQLSTATE values ***NONE (default)** Only an SQLSTATE value of 00000 is considered successful. *SQL-states* A list of one to ten allowable SQLSTATE values. Separate the SQLSTATE values with spaces. An SQLSTATE value is a five-digit hexadecimal number ranging from 00000 to FFFFF. The SQL statement is successful if it completes with an SQLSTATE value of 00000 or with one of the allowable SQLSTATE values that you listed.

Examples for ADDDPRSUB

The following examples illustrate how to use the **ADDDPRSUB** command.

Example 1:

To create a subscription set named SETHR under the AQHR Apply qualifier:

```
ADDDPRSUB APYQUAL(AQHR) SETNAME(SETHR) SRCTBL(HR/EMPLOYEE)
  TGTTBL(TGTLIB/TGTEMPL)
```

This subscription set, which contains one subscription-set member, replicates data from the registered source table named EMPLOYEE under the HR library to the target table named TGTEMPL under the TGTLIB library.

Example 2:

To create a subscription set named SETHR with only two columns, EMPNO (the key) and NAME, from the registered source table named EMPLOYEE and replicate these columns to an existing target table named TGTEMPL:

```
ADDDPRSUB APYQUAL(AQHR) SETNAME(SETHR) SRCTBL(HR/EMPLOYEE)
  TGTTBL(TGTLIB/TGTEMPL) CRTTGTTBL(*NO) COLUMN(EMPNO NAME) KEYCOL(EMPNO)
```

Example 3:

To create a subscription set named SETHR with data from the registered source table named EMPLOYEE and to replicate this data to a replica type target table named TGTREPL:

```
ADDDPRSUB APYQUAL(AQHR) SETNAME(SETHR) SRCTBL(HR/EMPLOYEE)
  TGTTBL(TGTLIB/TGTREPL) TGTTYPE(*REPLICA)
```

Example 4:

To create a subscription set named NOMEM with no subscription-set members:

```
ADDDPRSUB APYQUAL(AQHR) SETNAME(NOMEM) SRCTBL(*NONE) TGTTBL(*NONE)
```

ADDDPRSUBM: Adding a DPR subscription-set member (System i)

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

```
►►──ADDDPRSUBM──APYQUAL──(──apply-qualifier──)──SETNAME──(──set-name──)──────────►

►─SRCTBL──(──library-name/file-name──)──TGTTBL──(──library-name/file-name──)──────►

►──────────────────────────────────────────────────────────────────────────────────►
  │              ┌─*LOCAL───┐     │  │              ┌─*LOCAL───┐     │
  └─CTLSVR──(────┴─rdb-name─┴──)──┘  └─SRCSVR──(────┴─rdb-name─┴──)──┘

►──────────────────────────────────────────────────────────────────────────────────►
  │                ┌─*USERCOPY─────┐       │  │               ┌─*ALL─────────┐     │
  └─TGTTYPE──(─────┼─*POINTINTIME──┼──)────┘  └─ROWSLT──(─────┴─WHERE-clause─┴──)──┘
                   ├─*BASEAGR──────┤
                   ├─*CHANGEAGR────┤
                   ├─*CCD──────────┤
                   └─*REPLICA──────┘

►──────────────────────────────────────────────────────────────────────────────────►
  │                 ┌─*YES─┐       │  │             ┌─*YES─┐       │
  └─CRTTGTTBL──(────┴─*NO──┴───)───┘  └─CHKFMT──(───┴─*NO──┴───)───┘
```

```
►─┬─────────────────────────────┬─┬──────────────────────────────────────────┬─►
  │              ┌─*NO──┐       │ │                ┌─*ALL──────────────┐     │
  └─TGTKEYCHG──(─┴─*YES─┴──)────┘ └─COLUMN──(──────┼─*NONE─────────────┼──)──┘
                                                   │  ┌────────────┐   │
                                                   │  ▼        (1) │   │
                                                   └──column-name──┴───┘

►─┬─────────────────────────┬─┬──────────────────────────────────────┬──────────►
  │           ┌─*YES─┐      │ │               ┌─*SRCTBL──────────┐   │
  └─UNIQUE──(─┴─*NO──┴──)───┘ └─KEYCOL──(─────┼─*RRN─────────────┼─)─┘
                                              ├─*NONE────────────┤
                                              │  ┌───────────┐   │
                                              │  ▼       (2) │   │
                                              └──column-name─┴───┘

►─┬──────────────────────────────────────────────────────────┬──────────────────►
  │           ┌─*COLUMN──────────────────────────────┐       │
  │           │  ┌─────────────────────────────────┐ │       │
  │           │  ▼                             (3) │ │       │
  └─TGTCOL──(─┴──(column-name new-name)────────────┴─┴──)────┘

►─┬─────────────────────────────────────────────────────────────┬───────────────►
  │            ┌─*NONE──────────────────────────────────┐       │
  │            │  ┌──────────────────────────────────┐  │       │
  │            │  ▼                              (4) │  │       │
  └─CALCCOL──(─┴──(column-name expression)───────────┴──┴──)────┘

►─┬─────────────────────────┬──────────────────────────────────────────────────►◄
  │           ┌─*NO──┐      │
  └─ADDREG──(─┴─*YES─┴──)───┘
```

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 37 lists the invocation parameters.

Table 37. ADDDPRSUBM command parameter definitions for System i

Parameter	Definition and prompts
APYQUAL	Specifies the Apply qualifier that identifies which Apply program processes this subscription set. Subscription sets under an Apply qualifier run in a separate job. This parameter is required. *apply-qualifier* The name of the Apply qualifier.
SETNAME	Specifies the name of the subscription set. This parameter is required. *set-name* The name of the subscription set. The subscription-set name that you enter must be unique for the specified Apply qualifier or the **ADDDPRSUBM** command produces an error. Because the Apply program handles the set of target tables as a group, when one target table fails for any reason, the entire set fails.

Table 37. ADDDPRSUBM command parameter definitions for System i (continued)

Parameter	Definition and prompts
SRCTBL	Specifies the name of the table that is the source for this subscription-set member. You must register this table at the Capture control server before this table can become a member of a subscription set. This parameter is required. *library-name/file-name* The qualified name of the source table.
TGTTBL	Specifies the name of the target table for this subscription-set member. The target table is automatically created if you set the **CRTTGTTBL** parameter to *YES and the target table does not already exist. This parameter is required. *library-name/file-name* The qualified name of the target table.
CTLSVR	Specifies the relational database name of the system that contains the Apply control tables. ***LOCAL (default)** The Apply control tables reside locally (on the machine from which you are running the **ADDDPRSUBM** command). *rdb-name* The name of the relational database where the Apply control tables reside. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this name.
SRCSVR	Specifies the relational database name of the system that contains the Capture control tables. ***LOCAL (default)** The source table is registered on the local machine (the machine from which you are running the **ADDDPRSUBM** command). *rdb-name* The name of the relational database where the Capture control tables reside. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this name.

Table 37. ADDDPRSUBM command parameter definitions for System i (continued)

Parameter	Definition and prompts
TGTTYPE	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. ***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 System i system columns that are present in the point-in-time target table. This value is not valid when a value of *RRN is specified on the **KEYCOL** parameter. The table that you specified with the **SRCTBL** parameter must be one of the following types: user database, point-in-time table, or consistent-change data (CCD). **Important:** If the target table already exists, DB2 DataPropagator for System i does not automatically journal changes to it. You must start journaling outside of DB2 DataPropagator for System i. ***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 System i system column (IBMSNAP_LOGMARKER), which identifies when a particular row was inserted or updated at the Capture control server. ***BASEAGR** The target table is a base aggregate table, which is a target table that contains data that is aggregated (calculated) from a source table. The source table for a base aggregate target must be either a user table or a point-in-time table. This target table must contain the IBMSNAP_HLOGMARKER and IBMSNAP_LLOGMARKER system timestamp columns. ***CHANGEAGR** The table is a change aggregate table, which is a target table that contains data that is aggregated (calculated) based on the contents of a change-data (CD) table. This target table is created with the IBMSNAP_HLOGMARKER and IBMSNAP_LLOGMARKER system timestamp columns.

Table 37. ADDDPRSUBM command parameter definitions for System i (continued)

Parameter	Definition and prompts
TGTTYPE (Continued)	***CCD** The table is a consistent-change data (CCD) table, which is a target table created from a join of data in the change-data (CD) table and the unit-of-work (UOW) table. A CCD table provides transaction-consistent data for the Apply program and must include the following columns: • IBMSNAP_INTENTSEQ • IBMSNAP_OPERATION • IBMSNAP_COMMITSEQ • IBMSNAP_LOGMARKER ***REPLICA** The target table is a replica table, which is used only for update-anywhere replication. The replica target table receives changes from the master source table, and changes to the replica target table are propagated back to the master source table. A replica table is automatically registered as a source table.
ROWSLT	Specifies the predicates to be placed in an SQL WHERE clause. The Apply program uses these predicates to determine which rows in the change-data (CD) table of the source to apply to the target table. Use this parameter if you want only a subset of the source changes to be replicated to the target table. ***ALL (default)** The Apply program applies all changes in the CD table to the target table. *WHERE-clause* The SQL WHERE clause that specifies which rows from the CD table the Apply program applies to the target table. Do not include the WHERE keyword; it is implied on this parameter. This WHERE clause must be valid on the data server you are using to run the clause. **Note:** The WHERE clause on this parameter is unrelated to any WHERE clauses specified on the SQLBEFORE or SQLAFTER parameters.

Table 37. ADDDPRSUBM command parameter definitions for System i (continued)

Parameter	Definition and prompts
`CRTTGTTBL`	Specifies whether the target table (or view) is created. **`*YES (default)`** Creates the target table (or view) if it does not exist. Otherwise, the existing table or view becomes the target, and the format of this existing table or view is checked if the value of the **CHKFMT** parameter is set to *YES. An additional index, with the values that you specified by the **UNIQUE** and **KEYCOL** parameters, is created for a target table if no such index currently exists. The command fails if an existing target table contains rows that violate the conditions of the additional index. **`*NO`** Does not create the target table or view. You must create the table or view with the correct attributes before starting the Apply program. If the table or view exists and you set **CHKFMT** to *YES, the **`ADDDPRSUBM`** command ensures that the format of the existing table matches the subscription-set definition that you set. If **CHKFMT** is *NO, you must ensure that the format of the existing table matches the subscription-set definition. **Important:** If the table or view already exists, DB2 DataPropagator for System i does not automatically journal changes to the existing object. You must start journaling outside of DB2 DataPropagator for System i.
`CHKFMT`	Specifies whether DB2 DataPropagator for System i checks the definition of the subscription-set member against the existing target table to ensure that the columns match. This parameter is ignored if the **CRTTGTTBL** parameter is *YES; this parameter is also ignored if the **CRTTGTTBL** parameter is set to *NO and the target table does not exist. **`*YES (default)`** DB2 DataPropagator for System i verifies that the columns defined for this subscription-set member match the columns in the target table. This command fails if a mismatch is detected. **`*NO`** DB2 DataPropagator for System i ignores differences between the subscription-set member and the existing target table. You must ensure that the target table is compatible with the subscription-set member.

Table 37. ADDDPRSUBM command parameter definitions for System i (continued)

Parameter	Definition and prompts
TGTKEYCHG	Specifies how the Apply program handles updates when changes occur in source columns that are part of the target key columns for the target table. This parameter works in conjunction with the **USEDELINS** parameter on the **ADDDPRREG** command: • If **USEDELINS** is YES and **TGTKEYCHG** is YES, updates are not allowed. • If **USEDELINS** is YES and **TGTKEYCHG** is NO, updates become delete and insert pairs. • If **USEDELINS** is NO and **TGTKEYCHG** is YES, the Apply program handles this condition with special logic. • If **USEDELINS** is NO and **TGTKEYCHG** is NO, the Apply program processes the changes as normal updates. ***NO (default)** Updates to the source table are staged by the Capture program and processed by the Apply program to the target table. ***YES** The Apply program updates the target table based on the before images of the target key column, meaning that the Apply program changes the predicate to the old values instead of the new.
COLUMN	Specifies the columns to be included in the target table. The column names must be unqualified. Choose the column names from the list of column names that you specified on the **CAPCOL** parameter when you registered the source table. If you set the **IMAGE** parameter to *BOTH when registering this table, you can specify before-image column names. The before-image column names are the original column names with a prefix. This prefix is the character that you specified in the **PREFIX** parameter of the **ADDDPRREG** command. ***ALL (default)** All of the columns that you registered in the source are included in the target table. ***NONE** No columns from the source table are included in the target table. You can use *NONE when you want only computed columns in the target table. This value is required if the **CALCCOL** parameter contains summary functions but no grouping is performed. *column-name* The names of up to 300 source columns that you want to include in the target table. Separate the column names with spaces.

Table 37. ADDDPRSUBM command parameter definitions for System i (continued)

Parameter	Definition and prompts
`UNIQUE`	Specifies whether the target table has unique keys as indicated by the **KEYCOL** parameter. **`*YES (default)`** The target table supports one net change per key; only one row exists in the target table for that key regardless of how many changes are made to the key. This value specifies that the table contains current data rather than a history of changes to the data. A condensed table includes no more than one row for each primary key value in the table and can be used to supply current information for a refresh. **`*NO`** The target table supports multiple changes per key. The changes are appended to the target table. This value specifies that the table contains a history of changes to the data rather than current data. A non-condensed table includes more than one row for each key value in the table and can be used to supply a history of changes to the data. A non-condensed table cannot supply current data for a refresh.

Table 37. ADDDPRSUBM command parameter definitions for System i (continued)

Parameter	Definition and prompts
KEYCOL	Specifies columns that describe the key of the target table. The column names must be unqualified. For *POINTINTIME, *REPLICA, and *USERCOPY target tables (as specified on the **TGTTYPE** parameter), you must identify one or more columns as the target key for the target table. This target key is used by the Apply program to identify each unique row that changes during change-capture replication. ***SRCTBL (default)** The key columns in the target table are the same as those in the source table. The **ADDDPRREG** command uses the key that is specified in the source table if the source table has a key. The following key columns are used: • Key columns that you defined through DDS when creating the table with the Create Physical File (**CRTPF**) command • Primary and unique keys that you defined with the CREATE TABLE and ALTER TABLE SQL statements • Unique keys that you defined with the CREATE INDEX SQL statements If you use a column as a key more than once and with different ordering, the target table key is defined with ascending order. ***RRN** The key column in the target table is the IBMQSQ_RRN column. The target table is created with an IBMQSQ_RRN column, and this column is used as the key. When the Apply program runs, if the source table is a user table and the target table is a point-in-time table or user copy, the IBMQSQ_RRN column in the target table is updated with the relative record number of the associated record in the source table. Otherwise, the IBMQSQ_RRN column in the target table is updated with the value of the IBMQSQ_RRN column in the source table. ***NONE** The target copy does not contain a target key. You cannot specify *NONE if the target table type is *POINTINTIME, *REPLICA, or *USERCOPY. *column-name* The names of the target columns that you want to use as the target key columns. You can specify up to 120 column names. Separate the column names with spaces.

Table 37. ADDDPRSUBM command parameter definitions for System i (continued)

Parameter	Definition and prompts
TGTCOL	Specifies the new names for all the columns that the Apply program updates in the target table. These names override the column names taken from the source table. The column names must be unqualified. If you specified a value of *NONE for the **COLUMN** parameter, do not use the **TGTCOL** parameter. Use this parameter to give more meaningful names to the target table columns. Specify each source column name and the name of the corresponding column on the target table. **`*COLUMN (default)`** The target columns are the same as the columns you specified in the **COLUMN** parameter. *`column-name`* The column names from the source table that you want to change at the target. You can list up to 300 column names. *`new-name`* The new names of the target columns. You can list up to 300 new column names. If you do not use this parameter, the name of the column on the target table will be the same as the source column name.
CALCCOL	Specifies the list of user-defined or calculated columns in the target table. The column names must be unqualified. Enclose each column name and expression pair in parenthesis. You must specify a column name for each SQL expression. If you want to define any column as an SQL expression without a GROUP BY clause, you must set the **COLUMN** parameter to *NONE. **`*NONE (default)`** No user-defined or calculated columns are included in the target table. *`column-name`* The column names of the user-defined or calculated columns in the target table. You can list up to 100 column names. *`expression`* The expressions for the user-defined or calculated columns in the target table. You can list up to 100 SQL column expressions.
ADDREG	Specifies whether the target table is automatically registered as a source table. Use this parameter to register CCD target type tables. **`*NO (default)`** The target table is not registered as a source table. DB2 DataPropagator for System i ignores this parameter value if the target type is *REPLICA. Replica target tables are always automatically registered as source tables. **`*YES`** The target table is registered as a source table. This command fails if you already registered the target table. Do not set this parameter to *YES if the target table type is *USERCOPY, *POINTINTIME, *BASEAGR, or *CHANGEAGR. If you set the **CRTTGTTBL** parameter to *NO, you must create the target table before attempting to register it as a source.

Examples for ADDDPRSUBM

The following examples illustrate how to use the **ADDDPRSUBM** command.

Example 1:

To add a subscription-set member to a subscription set named SETHR under the AQHR Apply qualifier:

```
ADDDPRSUBM APYQUAL(AQHR) SETNAME(SETHR) SRCTBL(HR/YTDTAX) TGTTBL(TGTHR/TGTTAX)
```

Example 2:

To add a subscription-set member with only two columns, AMOUNT and NAME, from the registered source table named YTDTAX and to replicate these columns to an existing target table named TGTTAX:

```
ADDDPRSUBM APYQUAL(AQHR) SETNAME(SETHR) SRCTBL(HR/YTDTAX) TGTTBL(TGTLIB/TGTTAX)
  CRTTGTTBL(*NO) COLUMN(AMOUNT NAME) CHKFMT(*YES)
```

This command verifies that the AMOUNT and NAME columns defined for this subscription-set member match the columns in the target table.

Example 3:

To add a subscription-set member to subscription set named SETHR and to replicate this data to a consistent-change data target table named TGTYTD:

```
ADDDPRSUBM APYQUAL(AQHR) SETNAME(SETHR) SRCTBL(HR/YTDTAX) TGTTBL(TGTLIB/TGTYTD)
  TGTTYPE(*CCD) ADDREG (*YES)
```

This command registers the target table as a source table for DB2 DataPropagator for System i.

ANZDPR: Operating the Analyzer (System i)

Use the Analyze DPR (**ANZDPR**) command to analyze a failure from a Capture or Apply program, to verify the setup of your replication configuration, or to obtain problem diagnosis and performance tuning information.

Run this command after you set up your replication configuration.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

Syntax

```
►►──ANZDPR──┬────────────────────────────────────┬──────────────────────────►
            │          ┌─*LOCAL──────────────┐   │
            │          │ ┌─────────────────┐ │   │
            │          │ ▼            (1)  │ │   │
            └─RDB──(───┴───rdb-name────────┴─┴─)─┘
```

```
►─┬──────────────────────────────────────────────────────┬──────────────────►
  │                ┌─*CURLIB──────┐ ┌─ANZDPR───┐          │
  └─OUTFILE──(─────┴─library-name─┴─┴─file-name─┴──)──────┘

►─┬─────────────────────────────────┬─┬───────────────────────────────┬────►
  │                                 │ │              ┌─3───────────┐  │
  │                ┌─*STANDARD─┐    │ └─CAPTRC──(────┴─no-of-days──┴──)─┘
  └─ANZLVL──(──────┼─*SIMPLE───┼──)─┘
                   └─*DETAILED─┘

►─┬─────────────────────────────────┬─┬─────────────────────────────────────┬►
  │               ┌─3───────────┐   │ │                ┌─3───────────┐      │
  └─APYTRC──(─────┴─no-of-days──┴─)─┘ └─APYTRAIL──(────┴─no-of-days──┴──)───┘

►─┬─────────────────────────────────┬─┬───────────────────────────────┬────►
  │               ┌─3───────────┐   │ │              ┌─3───────────┐  │
  └─SIGTBL──(─────┴─no-of-days──┴─)─┘ └─CAPMON──(────┴─no-of-days──┴──)─┘

►─┬──────────────────────────────────────────┬───────────────────────────────►
  │                ┌─*ALL─────────────┐      │
  └─APYQUAL──(─────┴─apply-qualifier──┴──)───┘

►─┬──────────────────────────────────────────┬───────────────────────────────►◄
  │                  ┌─*ALL──────────┐       │
  └─CAPCTLLIB──(─────┴─library-name──┴──)────┘
```

Notes:

1 You can specify up to 10 databases.

Table 38 lists the invocation parameters.

Table 38. ANZDPR command parameter definitions for System i

Parameter	Definition and prompts
RDB	Specifies the databases to be analyzed. **`*LOCAL (default)`** The database on your local system. *rdb-name* The RDB Directory Entry name, which indicates the database. You can enter up to 10 databases. If you want to analyze multiple databases including the database on your local system, make sure that *LOCAL is the first entry in the list. Also, verify that you can connect to all these databases from your current system.

Table 38. ANZDPR command parameter definitions for System i (continued)

Parameter	Definition and prompts
OUTFILE	Specifies the library and file name used to store the analyzer output. This command writes the output to an HTML file. **`*CURLIB (default)`** The current library. *`library-name`* The name of the library. **`ANZDPR (default)`** The output is written to an HTML file named `ANZDPR`. *`file-name`* The name of the HTML output file. If the file name already exists, the file is overwritten. If the file name does not exist, the command creates the file with attributes of RCDLEN(512) and SIZE(*NOMAX).
ANZLVL	Specifies the level of analysis to be reported. The level of analysis can be: ***STANDARD (default)** Generates a report that includes the contents of the control tables as well as Capture and Apply program status information. ***SIMPLE** Generates the information in the standard report but excludes subcolumn details. Use this option if you want to generate a smaller report that requires less system resources. ***DETAILED** Generates a report with the most complete analysis. The detailed report includes the information from the standard report in addition to subscription set information.
CAPTRC	Specifies the date range (0 to 30 days) of entries to be reported from the IBMSNAP_CAPTRACE table. The default is 3. *`no-of-days`* The number of days to be reported.
APYTRC	Specifies the date range (0 to 30 days) of entries to be reported from the IBMSNAP_APPLYTRACE table. The default is 3. *`no-of-days`* The number of days to be reported.
APYTRAIL	Specifies the date range (0 to 30 days) of entries to be reported from the IBMSNAP_APPLYTRAIL table. The default is 3. *`no-of-days`* The number of days to be reported.
SIGTBL	Specifies the date range (0 to 30 days) of entries to be reported from the IBMSNAP_SIGNAL table. The default is 3. *`no-of-days`* The number of days to be reported.
CAPMON	Specifies the date range (0 to 30 days) of entries to be reported from the IBMSNAP_CAPMON table. The default is 3. *`no-of-days`* The number of days to be reported.

Table 38. ANZDPR command parameter definitions for System i (continued)

Parameter	Definition and prompts
APYQUAL	Specifies the Apply qualifiers to be analyzed. ***ALL (default)** All Apply qualifiers are analyzed. *apply-qualifier* The name of the Apply qualifier to be analyzed. You can enter up to 10 Apply qualifiers.
CAPCTLLIB	Specifies the Capture schemas, which are the names of the Capture control libraries that you want to analyze. You can analyze a specific Capture control library, or you can choose the default of *ALL to analyze all the Capture control libraries. ***ALL (default)** All of the Capture control libraries will be analyzed. *library-name* The name of the specific Capture control library that you want to analyze.

Examples for ANZDPR

The following examples illustrate how to use the **ANZDPR** command.

Example 1:

To run the Analyzer on both your local database and a remote database named RMTRDB1 using a standard level of analysis:

```
ANZDPR RDB(*LOCAL RMTRDB1) OUTFILE(MYLIB/ANZDPR) ANZLVL(*STANDARD) CAPTRC(1)
  APYTRC(1) APYTRAIL(1) SIGTBL(1) CAPMON(1) APYQUAL(*ALL)
```

This example generates one day of entries from the IBMSNAP_CAPTRACE, IBMSNAP_APPLYTRACE, IBMSNAP_APPLYTRAIL, IBMSNAP_SIGNAL, and IBMSNAP_CAPMON tables for all Apply qualifiers and writes the output to an HTML file named ANZDPR in the library called MYLIB.

Example 2:

To run the Analyzer with all default values:

```
ANZDPR
```

CHGDPRCAPA: Changing DPR Capture attributes (System i)

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

```
►►──CHGDPRCAPA──┬──────────────────────────────────────┬─────────────────────────►
                │                ┌─ASN──────────┐      │
                └─CAPCTLLIB(─────┴─library-name─┴──)───┘

►─┬──────────────────────────────────────┬─┬────────────────────────────┬────────►
  │            ┌─*SAME───────────┐       │ │         ┌─*SAME─────┐      │
  └─RETAIN──(──┴─retention-limit─┴──)────┘ └─LAG──(──┴─lag-limit─┴──)───┘

►─┬──────────────────────────────────────┬───────────────────────────────────────►
  │            ┌─*SAME───────────┐       │
  └─FRCFRQ──(──┴─force-frequency─┴──)────┘

►─┬────────────────────────────────────────┬─┬─────────────────────────────────┬─►
  │              ┌─*SAME──────────┐        │ │            ┌─*SAME───────┐      │
  └─CLNUPITV──(──┴─prune-interval─┴──)─────┘ └─TRCLMT──(──┴─trace-limit─┴──)───┘

►─┬──────────────────────────────────────┬─┬──────────────────────────────────────┬─►
  │            ┌─*SAME─────────┐         │ │            ┌─*SAME────────────┐      │
  └─MONLMT──(──┴─monitor-limit─┴──)──────┘ └─MONITV──(──┴─monitor-interval─┴──)───┘

►─┬────────────────────────────────────┬─────────────────────────────────────────►◄
  │            ┌─*SAME────────┐        │
  └─MEMLMT──(──┴─memory-limit─┴──)─────┘
```

Table 39 lists the invocation parameters.

Table 39. CHGDPRCAPA command parameter definitions for System i

Parameter	Definition and prompts
CAPCTLLIB	Specifies the Capture schema, which is the name of the library in which the Capture control tables reside. **ASN (default)** The Capture control tables are in the ASN library. *library-name* The name of a library that contains the Capture control tables.

Table 39. CHGDPRCAPA command parameter definitions for System i (continued)

Parameter	Definition and prompts
RETAIN	Specifies the new retention limit, which is the number of minutes that data is retained in the change-data (CD), unit-of-work (UOW), 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. This value works with the **CLNUPITV** parameter value. When the **CLNUPITV** value is reached, the CD, UOW, IBMSNAP_SIGNAL, and IBMSNAP_AUTHTKN data is removed if this data is older than the retention limit. Ensure that the Apply intervals are set to copy changed information before the data reaches this **RETAIN** parameter value to prevent inconsistent data in your tables. If the data becomes inconsistent, the Apply program performs a full refresh. The default is 10 080 minutes (seven days). The maximum is 35000000 minutes. ***SAME (default)** This value is not changed. *retention-limit* The new retention limit value.
LAG	Specifies the new lag limit, which is the number of minutes that the Capture program can fall behind in processing before restarting. This value is stored in the LAG_LIMIT column of the IBMSNAP_CAPPARMS table. When the lag limit is reached (that is, when the timestamp of the journal entry is older than the current timestamp minus the lag limit), the Capture program initiates a cold start for the tables that it is processing for that journal. The Apply program then performs a full refresh to provide the Capture program with a new starting point. The default is 10 080 minutes (seven days). The maximum is 35000000 minutes. ***SAME (default)** This value is not changed. *lag-limit* The new lag limit value.

Table 39. CHGDPRCAPA command parameter definitions for System i (continued)

Parameter	Definition and prompts
FRCFRQ	Specifies how often (from 30 to 600 seconds) the Capture program writes changes to the change-data (CD) and UOW tables. This value is stored in the COMMIT_INTERVAL column of the IBMSNAP_CAPPARMS table. The Capture program makes these changes available to the Apply program either when the buffers are filled or when the **FRCFRQ** time limit expires, whichever is sooner. Use this parameter to make changes more readily available to the Apply program on servers with a low rate of source table changes. The **FRCFRQ** parameter value is a global value used for all defined source tables. Setting the **FRCFRQ** value to a low number can affect system performance. The default is 30 seconds. ***SAME (default)** This value is not changed. *force-frequency* The new commit interval value, which is the number of seconds that the Capture program keeps CD and UOW table changes in buffer space before making these changes available to the Apply program.
CLNUPITV	Specifies the maximum amount of time (in hours) before the Capture program prunes old records from the change-data (CD), UOW, IBMSNAP_SIGNAL, IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_AUTHTKN tables. This parameter works in conjunction with the **RETAIN** parameter to control pruning of the CD, UOW, IBMSNAP_SIGNAL, and IBMSNAP_AUTHTKN tables, with the **MONLMT** parameter to control pruning of the IBMSNAP_CAPMON table, and with the **TRCLMT** parameter to control pruning of the IBMSNAP_CAPTRACE table. (Use the **STRDPRCAP** command to set the **RETAIN**, **MONLMT**, and **TRCLMT** parameters for a Capture program.) The value of this parameter is automatically converted from hours to seconds and is stored in the PRUNE_INTERVAL column of the IBMSNAP_CAPPARMS table. If the PRUNE_INTERVAL column is changed manually (not by using the **CHGDPRCAPA** command), you might see changes because of rounding when you prompt by using the F4 key. ***SAME (default)** This Capture attribute value is not changed. *prune-interval* The pruning interval expressed as a specific number of hours (1 to 100).

Table 39. CHGDPRCAPA command parameter definitions for System i (continued)

Parameter	Definition and prompts
TRCLMT	Specifies the trace limit (in minutes). This value is stored in the TRACE_LIMIT column of the IBMSNAP_CAPPARMS table. The Capture programs prune any IBMSNAP_CAPTRACE rows that are older than the trace limit. The default is 10 080 minutes (seven days of trace entries). ***SAME (default)** This value is not changed. *trace-limit* The number of minutes of trace data kept in the IBMSNAP_CAPTRACE table after pruning.
MONLMT	Specifies the monitor limit (in minutes). This value is stored in the MONITOR_LIMIT column of the IBMSNAP_CAPPARMS table. The Capture program prunes any IBMSNAP_CAPMON rows that are older than the monitor limit. The default is 10 080 minutes (seven days of monitor entries). ***SAME (default)** This value is not changed. *monitor-limit* The number of minutes of monitor data kept in the IBMSNAP_CAPMON table after pruning.
MONITV	Specifies how frequently (in seconds) the Capture program inserts rows into the IBMSNAP_CAPMON table. This value is stored in the MONITOR_INTERVAL column of the IBMSNAP_CAPPARMS table. The default is 300 seconds (five minutes). ***SAME (default)** This value is not changed. *monitor-interval* The number of seconds between row insertion into the IBMSNAP_CAPMON table. The monitor interval must be at least 120 seconds (two minutes). If you specify a number that is less than 120, this command automatically sets this parameter value to 120.
MEMLMT	Specifies the maximum size (in megabytes) of memory that the Capture journal job can use. This value is stored in the MEMORY_LIMIT column of the IBMSNAP_CAPPARMS table. The default is 32 megabytes. ***SAME (default)** This value is not changed. *memory-limit* The maximum number of megabytes for memory.

Examples for CHGDPRCAPA

The following examples illustrate how to use the **CHGDPRCAPA** command.

Example 1:

To change the frequency of row insertion to 6 000 seconds (100 minutes) by the Capture program into the IBMSNAP_CAPMON table:

```
CHGDPRCAPA CAPCTLLIB(ASN) MONITV(6000)
```

This frequency value is stored in the IBMSNAP_CAPPARMS table that is located in the default ASN library.

Example 2:

To change the retention limit, lag limit, trace limit, and monitor limit in the IBMSNAP_CAPPARMS table located in a Capture control library called LIB1:

```
CHGDPRCAPA CAPCTLLIB(LIB1) RETAIN(6000) LAG(3000) TRCLMT(3000) MONLMT(6000)
```

Example 3:

To change the commit interval, which indicates how frequently the Capture program writes changes to the CD and UOW tables:

```
CHGDPRCAPA CAPCTLLIB(ASN) FRCFRQ(360)
```

CRTDPRTBL: Creating the replication control tables (System i)

Use the Create DPR Tables **(CRTDPRTBL)** command to create replication control tables that are accidentally deleted or corrupted.

Important: The **CRTDPRTBL** command is the only command that you should use to create System i control tables. Do not use the Replication Center or ASNCLP command-line program to create the control tables.

Restriction: If you create an alternate Capture schema, you must created it in the same Auxiliary Storage Pool (either base or independent) where the ASN library is located.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

Syntax

```
►►──CRTDPRTBL──┬────────────────────────────────────┬──────────────────────────►◄
               │                 ┌─ASN──────────┐   │
               └─CAPCTLLIB──(────┴─library-name─┴──)┘
```

Table 40 on page 337 lists the invocation parameters.

Table 40. CRTDPRTBL command parameter definitions for System i

Parameter	Definition and prompts
CAPCTLLIB	Specifies the Capture schema, which is the name of the library where the newly created Capture control tables are placed. **ASN (default)** The Capture control tables are placed in the ASN library. *library-name* The name of the library where the Capture control tables are placed.

Examples for CRTDPRTBL

The following examples illustrate how to use the **CRTDPRTBL** command.

Example 1:

To create new replication control tables in the default ASN library:

```
CRTDPRTBL CAPCTLLIB(ASN)
```

Example 2:

To create new replication control tables for a Capture schema called DPRSALES:

```
CRTDPRTBL CAPCTLLIB(DPRSALES)
```

ENDDPRAPY: Stopping Apply (System i)

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.

```
►►──ENDDPRAPY──┬────────────────────────────┬──┬─────────────────────────┬──────────►
               │        ┌─*CURRENT──┐       │  │         ┌─*CNTRLD─┐     │
               └─USER(──┴─user-name─┴──)────┘  └─OPTION(──┴─*IMMED──┴──)──┘

►──┬──────────────────────────────────┬──┬───────────────────────────┬─────────►◄
   │          ┌─*USER───────────┐     │  │          ┌─*LOCAL───┐     │
   └─APYQUAL(─┴─apply-qualifier─┴──)──┘  └─CTLSVR(──┴─rdb-name─┴──)──┘
```

Table 41 on page 338 lists the invocation parameters.

Table 41. ENDDPRAPY command parameter definitions for System i

Parameter	Definition and prompts
USER	This parameter is ignored unless the **APYQUAL** parameter has a value of *USER, in which case this parameter specifies the Apply qualifier associated with the Apply program. ***CURRENT (default)** The Apply program of the user associated with the current job. *user-name* The Apply program of the specified user. When prompting on the **ENDDPRAPY** command, you can press the F4 key to see a list of users who defined subscriptions.
OPTION	Specifies how to stop the Apply program. ***CNTRLD (default)** The Apply program completes all of its tasks before stopping. These tasks might take a considerable period of time if the Apply program is completing a subscription set. ***IMMED** The Apply program completes all of its tasks with the **ENDJOB OPTION(*IMMED)** command. The tasks end immediately, without any cleanup. Use this option only after a controlled end is unsuccessful, because it can cause undesirable results. (Unless the Apply program was asleep when you issued the **ENDDPRAPY** command, you should verify the target table contents.) If the Apply program was performing a full refresh to the target table, the target table might be empty as a result of ending the Apply program before the table was refreshed with the source table contents. If the target table is empty, you must force a full refresh for this replication target. You might find that a subscription set is considered IN USE (the STATUS column in the IBMSNAP_SUBS_SET table has a value of 1). If it is, reset the value to 0 or -1. This allows the subscription set to be run again by the Apply program.
APYQUAL	Specifies the Apply qualifier that is used by the Apply program. ***USER (default)** The user name specified on the **USER** parameter is the Apply qualifier. *apply-qualifier* The name used to group the subscription sets that this Apply program runs. You can specify a maximum of 18 characters for the Apply qualifier name. This name follows the same naming conventions as a relational database name. You identify the subscriptions being run by the records in the IBMSNAP_SUBS_SET table with this value in the APPLY_QUAL column. When prompting on the **ENDDPRAPY** command, you can press the F4 key to see a list of Apply qualifier names with existing subscriptions.

Table 41. ENDDPRAPY command parameter definitions for System i (continued)

Parameter	Definition and prompts
`CTLSVR`	Specifies the relational database name of the system that contains the Apply control tables. **`*LOCAL (default)`** The Apply control tables reside locally (from the machine on which you are running the **`ENDDPRAPY`** command). *`rdb-name`* The name of the relational database where the Apply control tables reside. You can use the Work with RDB Directory Entries (**`WRKRDBDIRE`**) command to find this name. When prompting on the **`ENDDPRAPY`** command, you can press the F4 key to choose from the list of databases in the RDB directory.

Usage notes

The **`ENDDPRAPY`** command uses the value of the **APYQUAL** and **CTLSVR** parameters to search the 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 (System i)

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

```
►►──ENDDPRCAP──┬──────────────────────────┬──────────────────────────────────────►
               │            ┌─*CNTRLD─┐   │
               └─OPTION(────┴─*IMMED──┴──)┘

►─┬────────────────────────────────────┬─┬──────────────────────────────┬───────►◄
  │              ┌─ASN──────────┐      │ │               ┌─*NO──┐       │
  └─CAPCTLLIB──(─┴─library-name─┴──)───┘ └─RGZCTLTBL──(──┴─*YES─┴──)────┘
```

Table 42 lists the invocation parameters.

Table 42. ENDDPRCAP command parameter definitions for System i

Parameter	Definition and prompts
OPTION	Specifies how to stop the Capture program. ***CNTRLD (default)** The Capture program stops normally after completing all tasks. The **ENDDPRCAP** command might take longer when you specify the ***CNTRLD** option because the Capture program completes all of its subordinate processes before stopping. ***IMMED** The Capture program stops normally after completing all tasks with the **ENDJOB OPTION(*IMMED)** command.
CAPCTLLIB	Specifies the Capture schema, which is the name of the library in which the Capture control tables are located. This library includes the IBMSNAP_REGISTER table, which stores the registration information of the source tables. **ASN (default)** The Capture control tables are in the ASN library. The ASN library is the default library. *library-name* The name of a library that contains the Capture control tables.

Table 42. ENDDPRCAP command parameter definitions for System i (continued)

Parameter	Definition and prompts
RGZCTLTBL	Specifies whether a Reorganize Physical File Member (**RGZPFM**) command is performed on the control tables (including the change-data (CD) and unit-of-work (UOW) tables) when the Capture program ends. The system does not recover disk space unless the **RGZPFM** command process is performed on the tables. The **RGZPFM** command will not be performed if the control tables are being accessed by an Apply program or by other application programs. ***NO (default)** The **RGZPFM** command is not performed. ***YES** The **RGZPFM** command is performed.

Usage notes

If you use the **ENDJOB** command, temporary objects might be left in the QDP4 library. These objects have the types *DTAQ and *USRSPC, and are named QDP4*nnnnnn*, where *nnnnnn* is the job number of the job that used them. You can delete these objects when the job that used them (identified by the job number in the object name) is not active.

If the job under the Capture control library will not end after issuing this command, use the **ENDJOB** command with *IMMED option to end this job and all the journal jobs running in the DB2 DataPropagator for System i subsystem. Do not end Apply jobs running in the same subsystem if you want to end only the Capture program.

In rare cases when the Capture control job ends abnormally, the journal jobs created by Capture control job (which is named according to the **CAPCTLLIB** parameter) might still be left running. The only way to end these jobs is to use the **ENDJOB** command with either the *IMMED or *CNTRLD option.

Examples for ENDDPRCAP

The following examples illustrate how to use the **ENDDPRCAP** command.

Example 1:

To end the Capture program, which uses Capture control tables in the ASN library, after all processing tasks are completed:

```
ENDDPRCAP OPTION(*CNTRLD) CAPCTLLIB(ASN) RGZCTLTBL(*NO)
```

Example 2:

To end the Capture program immediately for the Capture schema BSN:

```
ENDDPRCAP OPTION(*IMMED) CAPCTLLIB(BSN) RGZCTLTBL(*NO)
```

Example 3:

To end the Capture program after all processing tasks are completed and to reorganize the Capture control tables:

```
ENDDPRCAP OPTION(*CNTRLD) CAPCTLLIB(ASN) RGZCTLTBL(*YES)
```

GRTDPRAUT: Authorizing users (System i)

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.

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

```
►►──GRTDPRAUT──┬───────────────────────────────────────┬─────────────────────►
               │                  ┌─ASN──────────┐     │
               └─CAPCTLLIB──(─────┴─library-name─┴──)──┘

                                         ┌─*REGISTRAR──┐
►─USER(─┬──────user-name────┬─)─AUT(─┼─*SUBSCRIBER─┼──)─────────────────────►
        └─*PUBLIC───────────┘            ├─*CAPTURE────┤
                                         └─*APPLY──────┘

            ┌─*ALL────────────┐
►─APYQUAL(──┼─*USER───────────┼──)────────────────────────────────────────────►◄
            └─apply-qualifier─┘
```

Table 43 lists the invocation parameters.

Table 43. GRTDPRAUT command parameter definitions for System i

Parameter	Definition and prompts
CAPCTLLIB	Specifies the Capture schema, which is the library that contains the replication control tables to which the user is granted authority. **ASN (default)** The Capture control tables reside in the ASN library. *library-name* The name of the library that contains the replication control tables.
USER	Specifies the users who have authority. *user-name* The names of up to 50 users who have authority. ***PUBLIC** Indicates that *PUBLIC authority is granted to the file, but (if insufficient for the task) is used only for those users who have no specific authority, who are not on the authorization list associated with the file, and whose group profile does not have any authority.

Table 43. GRTDPRAUT command parameter definitions for System i (continued)

Parameter	Definition and prompts
AUT	Specifies the type of authority being granted. ***REGISTRAR (default)** The users are granted the authorities to define, change, and remove registrations. For a complete list of authorities with AUT(*REGISTRAR), see Table 44 on page 344. ***SUBSCRIBER** The users are granted authority to define, change, and remove subscription sets. For a complete list of authorities with AUT(*SUBSCRIBER), see Table 45 on page 345. ***CAPTURE** The users are granted authority to run the Capture program. For a complete list of authorities granted with AUT(*CAPTURE), see Table 46 on page 346. ***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 a System i 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 47 on page 348.

Table 43. GRTDPRAUT command parameter definitions for System i (continued)

Parameter	Definition and prompts
APYQUAL	Specifies the Apply qualifier to be used by the user as specified with the **USER** parameter. This parameter is used only when AUT(*APPLY) or AUT(*SUBSCRIBER) is specified. ***ALL (default)** The user is granted authority to run the Apply program or to define and remove subscription sets for all Apply qualifiers. ***USER** The users specified on the **USER** parameter are granted authority to the subscription sets with an Apply qualifier that is the same as the user name. *apply-qualifier* The user is granted authority to run the Apply program or define and remove subscription sets for the Apply qualifiers associated with this Apply qualifier. • The user is granted authority to all replication sources, change-data (CD) tables, and consistent-change data (CCD) tables associated with records in the 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 44. Authorities granted with GRTDPRAUT AUT(*REGISTRAR)

Library	Object	Type	Authorizations
QSYS	capctllib	*LIB	*USE, *ADD
capctllib[1]	QSQJRN	*JRN	*OBJOPR, *OBJMGT
capctllib[1]	QZS8CTLBLK	*USRSPC	*CHANGE
capctllib[1]	IBMSNAP_REGISTER	*FILE	*OBJOPR, *READ, *ADD, *UPDT, *DLT

Table 44. 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 45. 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 45. 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 46. 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 46. 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 47. 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 47. Authorities granted with GRTDPRAUT AUT(*APPLY) (continued)*

Library	Object	Type	Authorizations
ASN	IBMSNAP_APPLYTRACX	*FILE	*OBJOPR, *READ, *UPD, *EXECUTE
ASN	IBMSNAP_SUBS_COLS	*FILE	*USE
ASN	IBMSNAP_SUBS_EVENT	*FILE	*USE
ASN	IBMSNAP_SUBS_STMTS	*FILE	*USE
ASN	IBMSNAP_SUBS_MEMBR	*FILE	*USE
ASN	ASN4A*	*SQLPKG	*USE
ASN	ASN4U*	*SQLPKG	*USE
ASN	IBMSNAP_APPLY_JOB	*FILE	*OBJOPR, *READ, *UPD, *ADD, *EXECUTE

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.

Examples for GRTDPRAUT

The following examples illustrate how to use the **GRTDPRAUT** command.

Example 1:

To authorize a user named USER1 to define and modify registrations:

```
GRTDPRAUT CAPCTLLIB(ASN) USER(USER1) AUT(*REGISTRAR)
```

Example 2:

To authorize a user named USER1 to define and modify subscription sets:

```
GRTDPRAUT CAPCTLLIB(ASN) USER(USER1) AUT(*SUBSCRIBER)
```

Example 3:

To authorize a user named USER1 to run Capture programs:

```
GRTDPRAUT CAPCTLLIB(ASN) USER(USER1) AUT(*CAPTURE)
```

Example 4:

To authorize a user named USER1 to define and modify existing subscription sets that are associated with Apply qualifier A1:

```
GRTDPRAUT CAPCTLLIB(ASN) USER(USER1) AUT(*SUBSCRIBER) APYQUAL(A1)
```

Example 5:

To authorize a user to run the Apply program on the control server system for all subscription sets associated with Apply qualifier A1, where the target server is the same as the control server:

1. Run the following command on the system where the Apply program will run:
   ```
   GRTDPRAUT CAPCTLLIB(ASN) USER(USER1) AUT(*APPLY) APYQUAL(A1)
   ```
2. Run the appropriate **GRTDPRAUT** command on the source server system:

- If the application server job on the source server used by the Apply program runs under user profile USER1, run the following command on the source server systems:

  ```
  GRTDPRAUT CAPCTLLIB(ASN) USER(USER1) AUT(*APPLY) APYQUAL(A1)
  ```
- If the application server job on the source server used by the Apply program runs under a different user profile, for example, QUSER, the command is:

  ```
  GRTDPRAUT CAPCTLLIB(ASN) USER(QUSER) AUT(*APPLY) APYQUAL(A1)
  ```

INZDPRCAP: Reinitializing DPR Capture (System i)

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

```
►►──INZDPRCAP──┬─────────────────────────────────────┬──────────────────────►
               │                ┌─ASN──────────┐     │
               └─CAPCTLLIB──(───┴─library-name─┴──)──┘

►──┬───────────────────────────────────────────────────┬────────────────────►◄
   │      ┌─*ALL──────────────────────────────┐        │
   │      │   ┌───────────────────────────┐   │        │
   │      │   ▼                           │   │        │
   └─JRN(─┴───library-name/journal-name───┴───┴───)────┘
```

Table 48 lists the invocation parameters.

Table 48. INZDPRCAP command parameter definitions for System i

Parameter	Definition and prompts
CAPCTLLIB	Specifies the Capture schema, which is the name of the library in which the Capture control tables reside. **ASN (default)** The Capture control tables reside in the ASN library. The ASN library is the default library. *library-name* The name of a library that contains the Capture control tables.

Table 48. INZDPRCAP command parameter definitions for System i (continued)

Parameter	Definition and prompts
JRN	Specifies a subset of up to 50 journals that you want the Capture program to work with. The Capture program starts processing all the source tables that are currently journaled to this journal. ***ALL (default)** The Capture program works with all the journals. *library-name/journal-name* The qualified name of the journal that you want the Capture program to work with.

Examples for INZDPRCAP

The following examples illustrate how to use the **INZDPRCAP** command.

Example 1:

To initialize a Capture program using the QSQJRN journal under a library named TRAINING:

```
INZDPRCAP CAPCTLLIB(ASN) JRN(TRAINING/QSQJRN)
```

The Capture control tables reside in the default ASN schema.

Example 2:

To initialize a Capture program that works with all the journals:

```
INZDPRCAP CAPCTLLIB(BSN) JRN(*ALL)
```

The Capture control tables reside in a schema called BSN.

OVRDPRCAPA: Overriding DPR Capture attributes (System i)

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

```
                           ┌─ASN──────────┐
►►──OVRDPRCAPA──CAPCTLLIB──(──┴─library-name─┴──)─────────────────────────►
```

```
►─┬──────────────────────────────────────────┬─┬─────────────────────────────────────────────┬──────────►
  │                ┌─*SAME───────────┐       │ │              ┌─*SAME────────────┐          │
  └─RETAIN──(──────┴─retention-limit─┴──)────┘ └─FRCFRQ──(────┴─force-frequency──┴──)───────┘

►─┬──────────────────────────────────────────┬─┬─────────────────────────────────────────────┬──────────►
  │                  ┌─*SAME──────────┐      │ │              ┌─*SAME───────┐               │
  └─CLNUPITV──(──────┴─prune-interval─┴──)───┘ └─TRCLMT──(────┴─trace-limit─┴──)────────────┘

►─┬──────────────────────────────────────────┬─┬─────────────────────────────────────────────┬──────────►
  │                ┌─*SAME─────────┐         │ │              ┌─*SAME────────────┐          │
  └─MONLMT──(──────┴─monitor-limit─┴──)──────┘ └─MONITV──(────┴─monitor-interval─┴──)───────┘

►─┬──────────────────────────────────────────┬─┬─────────────────────────────────────────────┬─────────►◄
  │                ┌─*SAME────────┐          │ │            ┌─*SAME─────┐                   │
  └─MEMLMT──(──────┴─memory-limit─┴──)───────┘ └─PRUNE──(───┼─*IMMED────┼──)───────────────┘
                                                            ├─*DELAYED──┤
                                                            └─*NO───────┘
```

Table 49 lists the invocation parameters.

Table 49. OVRDPRCAPA command parameter definitions for System i

Parameter	Definition and prompts
CAPCTLLIB	Specifies the Capture schema, which is the name of the library in which the Capture control tables reside. This library includes the IBMSNAP_REGISTER table, which stores the registration information of the source tables. This parameter is required. **ASN (default)** The Capture control tables reside in the ASN library. *library-name* The name of a library that contains the Capture control tables. You can create this library by using the **CRTDPRTBL** command with the **CAPCTLLIB** parameter.

Table 49. OVRDPRCAPA command parameter definitions for System i (continued)

Parameter	Definition and prompts
RETAIN	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. This value works with the **CLNUPITV** parameter value from the Start DPR Capture (**STRDPRCAP**) command. First, the Capture program deletes any CD, UOW, IBMSNAP_SIGNAL, or IBMSNAP_AUTHTKN rows that are older than the oldest currently running Apply program. Then, a new or remaining row from the CD, UOW, IBMSNAP_SIGNAL, or IBMSNAP_AUTHTKN table is subsequently deleted when its age reaches the value of the **RETAIN** parameter. Ensure that the Apply intervals are set to copy changed information before the data reaches this **RETAIN** parameter value to prevent inconsistent data in your tables. If the data becomes inconsistent, the Apply program performs a full refresh. The default is 10 080 minutes (seven days). The maximum is 35000000 minutes. ***SAME (default)** This value is not changed. *retention-limit* The new retention limit value.
FRCFRQ	Specifies how often (from 30 to 600 seconds) the Capture program writes changes to the change-data (CD) and unit-of-work (UOW) tables. The Capture program makes these changes available to the Apply program either when the buffers are filled or when the **FRCFRQ** time limit expires, whichever is sooner. This parameter value affects the amount of time that it takes for the Capture program to respond to changes from the Initialize DPR Capture (**INZDPRCAP**) command. Use this parameter to make changes more readily available to the Apply program on servers with a low rate of source table changes. The **FRCFRQ** parameter value is a global value used for all registered source tables. Setting the **FRCFRQ** value to a low number can affect system performance. The default is 30 seconds. ***SAME (default)** This value is not changed. *force-frequency* The new number of seconds that the Capture program keeps CD and UOW table changes in buffer space before making these changes available to the Apply program.

Table 49. OVRDPRCAPA command parameter definitions for System i (continued)

Parameter	Definition and prompts
`CLNUPITV`	Specifies the maximum amount of time (in hours) before the Capture program prunes old records from the change-data (CD), unit-of-work (UOW), IBMSNAP_SIGNAL, IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_AUTHTKN tables. This parameter works with the **RETAIN** parameter to control pruning of the CD, UOW, IBMSNAP_SIGNAL, and IBMSNAP_AUTHTKN tables, with the **MONLMT** parameter to control pruning of the IBMSNAP_CAPMON table, and with the **TRCLMT** parameter to control pruning of the IBMSNAP_CAPTRACE table. (Use the **`STRDPRCAP`** command to set the **RETAIN**, **MONLMT**, and **TRCLMT** parameters for a Capture program.) The value of the **CLNUPITV** parameter is automatically converted from hours to seconds and is stored in the PRUNE_INTERVAL column of the IBMSNAP_CAPPARMS table. **`*SAME (default)`** This Capture attribute value is not changed. *`prune-interval`* The pruning interval expressed as a specific number of hours (1 to 100).
`TRCLMT`	Specifies the trace limit, which indicates how frequently the IBMSNAP_CAPTRACE table is pruned. **`*SAME (default)`** The Capture program continues and uses the current trace limit value. *`trace-limit`* The number of minutes between each pruning operation of the IBMSNAP_CAPTRACE table.
`MONLMT`	Specifies the monitor limit, which indicates how frequently the IBMSNAP_CAPMON table is pruned. **`*SAME (default)`** The Capture program continues and uses the current monitor limit value. *`monitor-limit`* The number of minutes between each pruning operation of the IBMSNAP_CAPMON table.
`MONITV`	Specifies the monitor interval (in seconds), which indicates how frequently the Capture program inserts rows into the IBMSNAP_CAPMON table. **`*SAME (default)`** The Capture program continues and uses the current monitor interval value. *`monitor-interval`* The number of seconds between row insertion into the IBMSNAP_CAPMON table. The monitor interval must be at least 120 seconds (two minutes). If you type a number that is less than 120, the command automatically sets this parameter value to 120.

Table 49. OVRDPRCAPA command parameter definitions for System i (continued)

Parameter	Definition and prompts
MEMLMT	Specifies the maximum size (in megabytes) of memory that the Capture journal job can use. ***SAME (default)** The Capture program continues and uses the current memory limit value. *memory-limit* The maximum number of megabytes for memory.
PRUNE	Use this parameter to change the way that the Capture program prunes rows from the change-data (CD), unit-of-work (UOW), IBMSNAP_SIGNAL, IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_AUTHTKN tables. ***SAME (default)** The Capture program continues and uses the pruning parameters that you specified when you started the **STRDPRCAP** command. ***IMMED** The Capture program starts pruning the tables immediately, regardless of the value of the **CLNUPITV** parameter that you specified when you started the **STRDPRCAP** command. ***DELAYED** The Capture program prunes the old rows at the end of the specified pruning interval. PRUNE(*DELAYED) does not affect the frequency of pruning if you set the second part of the **CLNUPITV** parameter to *IMMED or *DELAYED on the **STRDPRCAP** command. However, PRUNE(*DELAYED) *does* initiate pruning if you set the second part of the **CLNUPITV** parameter to *NO when you started the **STRDPRCAP** command. ***NO** The Capture program does not initiate pruning. This value overrides the **CLNUPITV** parameter setting from the **STRDPRCAP** command.

Examples for OVRDPRCAPA

The following examples illustrate how to use the **OVRDPRCAPA** command.

Example 1:

To change the pruning parameters of the CD, UOW, IBMSNAP_SIGNAL, IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_AUTHTKN tables (which reside under the default ASN library) and to change the IBMSNAP_CAPMON monitor interval and memory limit of Capture journal jobs in a running Capture program:

```
OVRDPRCAPA CAPCTLLIB(ASN) CLNUPITV(12) MONITV(600) MEMLMT(64)
```

Example 2:

To initiate pruning of the CD, UOW, IBMSNAP_SIGNAL, IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_AUTHTKN tables, which reside in the BSN library:

```
OVRDPRCAPA CAPCTLLIB(BSN) PRUNE(*IMMED)
```

RMVDPRREG: Removing a DPR registration (System i)

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──)──────────────────────►

►─┬────────────────────────────────────┬──────────────────────────────►◄
  │                 ┌─ASN──────────┐    │
  └─CAPCTLLIB──(────┴─library-name─┴──)─┘
```

Table 50 lists the invocation parameters.

Table 50. RMVDPRREG command parameter definitions for System i

Parameter	Definition and prompts
SRCTBL	Identifies the registration that you want to remove. This is a required parameter. *library-name/file-name* The qualified name of the registered file.
CAPCTLLIB	Specifies the Capture schema, which is the name of the library in which the Capture control tables reside. **ASN (default)** The Capture control tables are in the ASN library. *library-name* The name of a library containing the Capture control tables.

Examples for RMVDPRREG

The following examples illustrate how to use the **RMVDPRREG** command.

Example 1:

To remove the registration for the source table named EMPLOYEE of the HR library in the default ASN Capture schema:

```
RMVDPRREG SRCTBL(HR/EMPLOYEE)
```

Example 2:

To remove the registration for the source table named SALES of the DEPT library under a Capture schema called BSN:

```
RMVDPRREG SRCTBL(DEPT/SALES) CAPCTLLIB(BSN)
```

RMVDPRSUB: Removing a DPR subscription set (System i)

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──)──────────►

►─┬──────────────────────────────────┬─┬─────────────────────────────┬───────────►
  │                ┌─*LOCAL───┐      │ │             ┌─*NO──┐        │
  └─CTLSVR──(──────┴─rdb-name─┴───)──┘ └─RMVREG──(────┴─*YES─┴───)───┘

►─┬────────────────────────────────┬─┬──────────────────────────────┬──────────►◄
  │                ┌─*NO──┐        │ │              ┌─*NO──┐        │
  └─DLTTGTTBL──(───┴─*YES─┴───)────┘ └─RMVMBRS──(───┴─*YES─┴───)────┘
```

Table 51 lists the invocation parameters.

Table 51. RMVDPRSUB command parameter definitions for System i

Parameter	Definition and prompts
APYQUAL	Specifies the Apply qualifier that the Apply program uses to identify the subscription set. This parameter is required. *apply-qualifier* The name of the Apply qualifier.
SETNAME	Specifies the name of the subscription set. This parameter is required. *set-name* The name of the subscription set. You receive an error message if you enter a subscription-set name that does not exist for the specified Apply qualifier.
CTLSVR	Specifies the relational database name of the system that contains the Apply control tables. ***LOCAL (default)** The Apply control tables reside locally (on the machine from which you are running the **RMVDPRSUB** command). *rdb-name* The name of the relational database where the Apply control tables reside. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this name.

Table 51. RMVDPRSUB command parameter definitions for System i (continued)

Parameter	Definition and prompts
RMVREG	Specifies whether this command removes the registrations that are associated with the target tables of all the subscription-set members in the subscription set. Use this parameter only if you have set the **RMVMBRS** parameter to *YES. **`*NO (default)`** The registrations are not removed. **`*YES`** The registrations are removed.
DLTTGTTBL	Specifies whether this command drops the target tables of the subscription-set members after the subscription set is removed. Use this parameter only if you set the **RMVMBRS** parameter to *YES. **`*NO (default)`** The target tables are not dropped. **`*YES`** The target tables are dropped.
RMVMBRS	Specifies whether this command removes the subscription set and all the members in that subscription set. **`*NO (default)`** The subscription set is not removed if there are existing members in the subscription set. **`*YES`** The subscription set and all its subscription-set members are removed.

Examples for RMVDPRSUB

The following examples illustrate how to use the **RMVDPRSUB** command.

Example 1:

To remove a subscription set named SETHR that contains no subscription-set members:

```
RMVDPRSUB APYQUAL(AQHR) SETNAME(SETHR)
```

Example 2:

To remove a subscription set named SETHR and all its subscription-set members:

```
RMVDPRSUB APYQUAL(AQHR) SETNAME(SETHR) RMVMBRS(*YES)
```

Example 3:

To remove a subscription set named SETHR, all its subscription-set members, and the associated registrations:

```
RMVDPRSUB APYQUAL(AQHR) SETNAME(SETHR) RMVREG(*YES) RMVMBRS(*YES)
```

RMVDPRSUBM: Removing a DPR subscription-set member (System i)

Use the Remove DPR subscription-set member (**`RMVDPRSUBM`**) command to remove a single subscription-set member from a subscription set.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

Syntax

```
►►──RMVDPRSUBM──APYQUAL──(──apply-qualifier──)──SETNAME──(──set-name──)──────────►

►─TGTTBL──(──library-name/file-name──)────────────────────────────────────────────►
                                         └─CTLSVR──(──┬─*LOCAL───┬──)─┘
                                                      └─rdb-name─┘

►──┬──────────────────────────────┬──┬──────────────────────────────────┬──────────►◄
   └─RMVREG──(──┬─*NO──┬──)─┘        └─DLTTGTTBL──(──┬─*NO──┬──)─┘
                └─*YES─┘                             └─*YES─┘
```

Table 52 lists the invocation parameters.

Table 52. RMVDPRSUBM command parameter definitions for System i

Parameter	Definition and prompts
APYQUAL	Specifies the Apply qualifier that the Apply program uses to identify the subscription set. This parameter is required. *apply-qualifier* The name of the Apply qualifier.
SETNAME	Specifies the name of the subscription set. This parameter is required. *set-name* The name of the subscription set. You receive an error message if you enter a subscription-set name that does not exist for the specified Apply qualifier.
TGTTBL	Specifies the target table that is registered for the subscription-set member. This parameter is required. *library-name/file-name* The qualified name of the target table.
CTLSVR	Specifies the relational database name of the system that contains the Apply control tables. ***LOCAL (default)** The Apply control tables reside locally (on the machine from which you are running the **RMVDPRSUBM** command). *rdb-name* The name of the relational database where the Apply control tables reside. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this name.
RMVREG	Specifies whether this command removes the registration that is associated with the target table for the subscription-set member. ***NO (default)** The registration is not removed. ***YES** The registration is removed.

Table 52. RMVDPRSUBM command parameter definitions for System i (continued)

Parameter	Definition and prompts
DLTTGTTBL	Specifies whether this command drops the target table of the subscription-set member after the subscription-set member is removed. ***NO (default)** The target table is not dropped. ***YES** The target table is dropped.

Examples for RMVDPRSUBM

The following examples illustrate how to use the **RMVDPRSUBM** command.

Example 1:

To remove a subscription-set member, which uses a target table named EMP, from the SETEMP subscription set on the relational database named RMTRDB1:

```
RMVDPRSUBM APYQUAL(AQHR) SETNAME(SETEMP) TGTTBL(TGTEMP/EMP) CTLSVR(RMTRDB1)
```

Example 2:

To remove a subscription-set member from the SETHR subscription set, remove the registration, and then drop the table:

```
RMVDPRSUBM APYQUAL(AQHR) SETNAME(SETHR) TGTTBL(TGTHR/YTDTAX) RMVREG(*YES)
  DLTTGTTBL(*YES)
```

RVKDPRAUT: Revoking authority (System i)

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

```
                                                        ┌──────────────────┐
                                                        ▼                  │
►►──RVKDPRAUT──────────────────────────────────USER(──┬───user-name──────┴──┬──)──────────►◄
             │                                    │   └──*PUBLIC───────────────┘
             │                ┌─ASN───────────┐  │
             └─CAPCTLLIB──(───┴─library-name──┴──)─┘
```

Table 53 on page 361 lists the invocation parameters.

Table 53. RVKDPRAUT command parameter definitions for System i

Parameter	Definition and prompts
CAPCTLLIB	Specifies the Capture schema, which is the name of the library under which user authority is being revoked. **ASN (default)** The Capture control tables reside in the ASN library. *library-name* The name of the library that contains the replication control tables.
USER	Specifies the users whose authority is revoked. This parameter is required. *user-name* Specifies the names of up to 50 users whose authority is revoked. ***PUBLIC** Specifies that authority is revoked from all users without specific authority, who are not on the authorization list, and whose group profile does not have any authority.

Usage notes

The command returns an error message if any of the following conditions exist:

- A specified user does not exist.
- The user running the command is not authorized to the specified user profiles.
- The user running the command does not have permission to revoke authorities to the DB2 DataPropagator for System i control tables.
- The DB2 DataPropagator for System i 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 (System i)

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.

```
►►──STRDPRAPY──┬──────────────────────────────┬────────────────────────────────────►
               │             ┌─*CURRENT──┐    │
               └─USER(───────┼─*JOBD─────┼──)─┘
                             └─user-name─┘

►──┬───────────────────────────────────────────────────┬───────────────────────────►
   │        ┌─*LIBL/QZSNDPR────────────────────┐       │
   └─JOBD(──┴─library-name/job-description-name┴──)────┘

►──┬──────────────────────────────────────┬──┬────────────────────────────┬─────────►
   │           ┌─*USER───────────┐        │  │          ┌─*LOCAL──┐       │
   └─APYQUAL(──┴─apply-qualifier─┴──)─────┘  └─CTLSVR(──┴─rdb-name┴──)──┘

►──┬──────────────────────────────┬─────────────────────────────────────────────────►
   │         ┌─*NONE────┐         │
   └─TRACE(──┼─*ERROR───┼──)──────┘
             ├─*ALL─────┤
             ├─*PRF─────┤
             └─*REWORK──┘

►──┬──────────────────────────────────────────────────────┬─────────────────────────►
   │               ┌─*NONE─────────────────────┐          │
   └─FULLREFPGM(───┴─library-name/program-name─┴──)───────┘

►──┬─────────────────────────────────────────────────┬──┬───────────────────────┬───►
   │              ┌─*NONE─────────────────────┐      │  │            ┌─*YES─┐   │
   └─SUBNFYPGM(───┴─library-name/program-name─┴──)───┘  └─INACTMSG(──┴─*NO──┴─)─┘

►──┬──────────────────────────┬──┬────────────────────────────┬─────────────────────►
   │            ┌─*YES─┐      │  │         ┌─6──────────┐     │
   └─ALWINACT(──┴─*NO──┴──)───┘  └─DELAY(──┴─delay-time─┴──)──┘

►──┬──────────────────────────────────────┬──┬──────────────────────────┬───────────►
   │           ┌─300──────────────┐       │  │            ┌─*NO──┐      │
   └─RTYWAIT(──┴─retry-wait-time──┴──)────┘  └─COPYONCE(──┴─*YES─┴──)───┘

►──┬──────────────────────────┬──┬──────────────────────────┬──────────────────────►◄
   │            ┌─*NO──┐      │  │             ┌─*NO──┐     │
   └─TRLREUSE(──┴─*YES─┴──)───┘  └─OPTSNGSET(──┴─*YES─┴──)──┘
```

Table 54 on page 363 lists the invocation parameters.

Table 54. STRDPRAPY command parameter definitions for System i

Parameter	Definition and prompts
USER	Specifies the name of the user ID for which to start the Apply program. When you run this command, you must be authorized (have *USE rights) to the specified user profile; the Apply program runs under this specified user profile. The control tables are located on the relational database specified by the **CTLSVR** parameter. The same control tables are used regardless of the value specified on the **USER** parameter. **`*CURRENT (default)`** The user ID associated with the current job is the same user ID associated with this Apply program. **`*JOBD`** The user ID specified in the job description associated with this Apply program. The job description cannot specify USER(*RQD). *`user-name`* The user ID associated with this Apply program. The following IBM-supplied objects are *not* valid on this parameter: QDBSHR, QDFTOWN, QDOC, QLPAUTO, QLPINSTALL, QRJE, QSECOFR, QSPL, QSYS, or QTSTRQS. When prompting on the **`STRDPRAPY`** command, you can press the F4 key to see a list of users who defined subscription sets.
JOBD	Specifies the name of the job description to use when submitting the Apply program. **`*LIBL/QZSNDPR (default)`** The default job description provided with DB2 DataPropagator for System i. *`library-name/job-description-name`* The name of the job description used for the Apply program.
APYQUAL	Specifies the Apply qualifier to be used by the Apply program. All subscriptions sets that are grouped together with this Apply qualifier are run by the Apply program. **`*USER (default)`** The **USER** parameter value that you enter is used as the name of the Apply qualifier. *`apply-qualifier`* The name used to group the subscription sets that are to be run by this Apply program. You can specify a maximum of 18 characters for the Apply qualifier name. This name follows the same naming conventions as a relational database name. When prompting on the **`STRDPRAPY`** command, you can press the F4 key to see a list of Apply qualifier names with existing subscription sets.

Table 54. STRDPRAPY command parameter definitions for System i (continued)

Parameter	Definition and prompts
CTLSVR	Specifies the relational database name of the system that contains the Apply control tables. ***LOCAL (default)** The Apply control tables reside locally (on the machine where you are running the **STRDPRAPY** command). *rdb-name* The name of the relational database where the Apply control tables reside. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this name. When prompting on the **STRDPRAPY** command, you can press the F4 key to see a list of available RDB names.
TRACE	Specifies whether the Apply program should generate a trace. The Apply program writes the trace data to a spool file called QPZSNATRC. ***NONE (default)** No trace is generated. ***ERROR** The trace contains error information only. ***ALL** The trace contains error and execution flow information. ***PRF** The trace contains information that can be used to analyze performance at different stages of the Apply program execution. ***REWORK** The trace contains information about rows that were reworked by the Apply program.
FULLREFPGM	Specifies whether the Apply program is to invoke an exit routine to initialize a target table. When the Apply program is ready to perform a full refresh of a target table, the Apply program invokes the specified exit routine rather than performing the full refresh itself. When a full-refresh exit routine is used by the Apply program, the value of the ASNLOAD column in the IBMSNAP_APPLYTRAIL table is Y. ***NONE (default)** A full-refresh exit routine is not used. *library-name/program-name* The qualified name of the program that is called by the Apply program performing a full refresh of a target table. For example, to call program ASNLOAD in library DATAPROP, the qualified name is DATAPROP/ASNLOAD.

Table 54. STRDPRAPY command parameter definitions for System i (continued)

Parameter	Definition and prompts
SUBNFYPGM	Specifies whether the Apply program is to invoke an exit routine when the program finishes processing a subscription set. Input to the exit routine includes the subscription set name, Apply qualifier, completion status, and statistics with the number of rejects. The notify program allows you to examine the unit-of-work (UOW) table to determine when transactions have been rejected and when to take further actions, such as issuing a message or generating an event. ***NONE (default)** An exit routine is not used. *library-name/program-name* The qualified name of the exit routine program called by the Apply program when processing a subscription set. For example, to call program APPLYDONE in library DATAPROP, the qualified name is DATAPROP/APPLYDONE.
INACTMSG	Specifies whether the Apply program is to generate a message whenever the program completes its work and becomes inactive for a period of time. ***YES (default)** The Apply program generates message ASN1044 before beginning a period of inactivity. Message ASN1044 indicates how long the Apply program remains inactive. ***NO** No message is generated.
ALWINACT	Specifies whether the Apply program can run in an inactive (sleep) state. ***YES (default)** The Apply program sleeps if there is nothing to process. ***NO** If the Apply program has nothing to process, the job that submitted and started the Apply program ends.
DELAY	Specifies the delay time (in seconds) at the end of each Apply program cycle when continuous replication is used. **6 (default)** The delay time is six seconds. *delay-time* The delay time, entered as a number between 0 and 6 inclusive.
RTYWAIT	Specifies how long (in seconds) the Apply program is to wait after it encounters an error before it retries the operation that failed. **300 (default)** The retry wait time is 300 seconds (five minutes). *retry-wait-time* The wait time, entered as a number between 0 and 35000000 inclusive, before the Apply program retries the failed operation.

Table 54. STRDPRAPY command parameter definitions for System i (continued)

Parameter	Definition and prompts
`COPYONCE`	Specifies whether the Apply program executes one copy cycle for each subscription set that is eligible at the time the Apply program is invoked. Then the Apply program terminates. An eligible subscription set meets the following criteria: • (ACTIVATE > 0) in the IBMSNAP_SUBS_SET table. When the ACTIVATE column value is greater than zero, the subscription set is active indefinitely or is used for a one-time-only subscription processing. • (REFRESH_TYPE = R or B) or (REFRESH_TYPE = E and the specified event occurred). The REFRESH_TYPE column value is stored in the IBMSNAP_SUBS_SET table. The MAX_SYNCH_MINUTES limit from the IBMSNAP_SUBS_SET table and the END_OF_PERIOD timestamp from the IBMSNAP_SUBS_EVENT table are honored if specified. ***NO (default)** The Apply program does not execute one copy cycle for each eligible subscription set. ***YES** The Apply program executes one copy cycle for each eligible subscription set and then terminates.
`TRLREUSE`	Specifies whether the Apply program empties the IBMSNAP_APPLYTRAIL table when the Apply program starts. ***NO (default)** The Apply program does not empty the IBMSNAP_APPLYTRAIL table during program startup. ***YES** The Apply program empties the IBMSNAP_APPLYTRAIL table during program startup.
`OPTSNGSET`	Specifies whether the performance of the Apply program is optimized if only one subscription set is processed. This parameter does not pertain to replica target tables. If you set this parameter to *YES, the Apply program fetches the members and columns of a subscription set only once and reuses this fetched information when processing the same subscription set in two or more consecutive processing cycles. ***NO (default)** The performance of the Apply program is not optimized if only one subscription set is processed. ***YES** The performance of the Apply program is optimized if only one subscription set is processed. The Apply program reuses the subscription set information during subsequent processing cycles, requiring fewer CPU resources and improving throughput rates.

Usage notes

You can set up the system to start the subsystem automatically by adding the command that is referred to in the QSTRUPPGM value on your system. If you use the QDP4/QZSNDPR subsystem, it is started as part of the **STRDPRAPY** command processing.

If the relational database (RDB) specified by the **CTLSVR** parameter is a DB2 for i5/OS database, the tables on the server are found in the ASN library. If the RDB is not a DB2 for i5/OS 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 by 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 System i 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.

Examples for STRDPRAPY

The following examples illustrate how to use the **STRDPRAPY** command.

Example 1:

To start the Apply program that uses the AQHR Apply qualifier and Apply control tables that reside locally and to generate a trace file that contains error and execution flow information:

```
STRDPRAPY APYQUAL(AQHR) CTLSVR(*LOCAL) TRACE(*ALL)
```

Example 2:

To start an Apply program with Apply control tables that reside locally and to specify that the job that started this Apply program automatically end when the Apply program has nothing left to process:

```
STRDPRAPY APYQUAL(AQHR) CTLSVR(*LOCAL) ALWINACT(*NO)
```

Example 3:

To start an Apply program that empties the IBMSNAP_APPLYTRAIL table during program startup:

```
STRDPRAPY APYQUAL(AQHR) CTLSVR(*LOCAL) TRLREUSE(*YES)
```

Example 4:

To start an Apply program with all default values:

```
STRDPRAPY
```

STRDPRCAP: Starting Capture (System i)

Use the Start DPR Capture (**STRDPRCAP**) command to start capturing changes to System i database tables on System i servers.

Because this command processes all replication sources in the IBMSNAP_REGISTER table, make sure that you are running this command with the proper authority.

After you start the Capture program, it runs continuously until you stop it or it detects an unrecoverable error.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

Syntax

```
►►──STRDPRCAP──┬──────────────────────────┬──────────────────────────────────►
               │             ┌─*YES─┐      │
               └─RESTART(────┴─*NO──┴───)──┘
```

```
►─┬───────────────────────────────────────────────────────┬──────────────►
  │           ┌─*LIBL/QZSNDPR────────────────────┐         │
  └─JOBD(─────┴─library-name/job-description-name─┴──)─────┘

►─┬──────────────────────────┬───────────────────────────────────────────►
  │           ┌─120───┐      │
  └─WAIT──(───┴─value─┴──)───┘

►─┬───────────────────────────────────────────────────────┬──────────────►
  │                ┌─*DFT──────────┐   ┌─*IMMED───┐        │
  └─CLNUPITV──(────┴─hours-to-wait─┴───┼─*DELAYED─┼──)─────┘
                                       └─*NO──────┘

►─┬──────────────────────────────────────┬───────────────────────────────►
  │                ┌─ASN──────────┐      │
  └─CAPCTLLIB──(───┴─library-name─┴──)───┘

►─┬──────────────────────────────────────────────────────────┬───────────►
  │           ┌─*ALL───────────────────────────────────┐      │
  │           │   ┌──────────────────────────────────┐ │      │
  │           │   ▼                              (1) │ │      │
  └─JRN──(────┴───library-name/journal-name──────────┴─┴──)───┘

►─┬──────────────────────────────────┬─┬──────────────────────────────────┬─►
  │            ┌─*DFT────────┐       │ │            ┌─*DFT──────────┐     │
  └─TRCLMT──(──┴─trace-limit─┴──)────┘ └─MONLMT──(──┴─monitor-limit─┴──)──┘

►─┬────────────────────────────────────────┬─┬──────────────────────────────────┬─►
  │            ┌─*DFT─────────────┐        │ │            ┌─*DFT─────────┐      │
  └─MONITV──(──┴─monitor-interval─┴──)─────┘ └─MEMLMT──(──┴─memory-limit─┴──)───┘

►─┬──────────────────────────────────────┬─┬──────────────────────────────┬─►
  │            ┌─*DFT────────────┐       │ │         ┌─*DFT──────┐        │
  └─RETAIN──(──┴─retention-limit─┴──)────┘ └─LAG──(──┴─lag-limit─┴──)─────┘

►─┬──────────────────────────────────────┬──────────────────────────────►◄
  │            ┌─*DFT────────────┐       │
  └─FRCFRQ──(──┴─force-frequency─┴──)────┘
```

Notes:

1 You can specify up to 50 journals.

Table 55 on page 370 lists the invocation parameters.

Table 55. STRDPRCAP command parameter definitions for System i

Parameter	Definition and prompts
RESTART	Specifies how the Capture program handles warm and cold starts. ***YES (default)** The Capture program continues processing the changes from the point where it was when it ended previously. This is also known as a *warm start* and is the normal mode of operation. ***NO** The Capture program removes all information from the change-data (CD) tables. The Capture program also removes all information from the unit-of-work (UOW) table when you specify JRN(*ALL). All subscriptions for affected source tables are fully refreshed before change capturing resumes. This process is also known as a *cold start*. By specifying RESTART(*NO) and JRN(*library-name/journal-name*), you can cold start the Capture program for specified journals.
JOBD	Specifies the name of the job description to use when submitting the Capture program. ***LIBL/QZSNDPR (default)** Specifies the default job description provided with DB2 DataPropagator for System i. *library-name/job-description-name* The name of the job description used for the Capture program.
WAIT	Specifies the maximum number of seconds (60 to 6 000) to wait before the Capture program checks its status. You can use this value to tune the responsiveness of the Capture program. A low value reduces the time that the Capture program takes before ending or initializing, but can have a negative effect on system performance. A higher value increases the time that the Capture program takes before ending or initializing, but can improve system performance. A value that is too high can result in decreased responsiveness while the Capture program is performing periodic processing. The amount of the decrease in responsiveness depends on the amount of change activity to source tables and the amount of other work occurring on the system. **120 (default)** The Capture program waits 120 seconds. *value* The maximum number of seconds that the Capture program waits.

Table 55. STRDPRCAP command parameter definitions for System i (continued)

Parameter	Definition and prompts
CLNUPITV	Specifies the maximum amount of time (in hours) before the Capture program prunes old records from the change-data (CD), unit-of-work (UOW), IBMSNAP_SIGNAL, IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_AUTHTKN tables. This parameter works with the **RETAIN** parameter to control pruning of the CD, UOW, IBMSNAP_SIGNAL, and IBMSNAP_AUTHTKN tables, with the **MONLMT** parameter to control pruning of the IBMSNAP_CAPMON table, and with the **TRCLMT** parameter to control pruning of the IBMSNAP_CAPTRACE table. (Use the **STRDPRCAP** command to set the **RETAIN**, **MONLMT**, and **TRCLMT** parameters for a Capture program. Use the **CHGDPRCAPA** or **OVRDPRCAPA** command to change these parameter settings.) There are two parts to the **CLNUPITV** parameter: ***DFT (default)** The Capture program uses the value of the PRUNE_INTERVAL column from the IBMSNAP_CAPPARMS table. *hours-to-wait* The pruning interval expressed as a specific number of hours (1 to 100). ***IMMED (default)** The Capture program prunes old records at the beginning of the specified interval (or immediately), and at each interval thereafter. ***DELAYED** The Capture program prunes old records at the end of the specified interval, and at each interval thereafter. ***NO** The Capture program does not prune records.
CAPCTLLIB	Specifies the Capture schema, which is the name of the library in which the Capture control tables reside. **ASN (default)** The default library in which the Capture control tables reside. *library-name* The name of the library in which the Capture control tables reside.
JRN	Specifies a subset of up to 50 journals that you want the Capture program to work with. The Capture program starts processing all the source tables that are currently journaled to this journal. ***ALL (default)** The Capture program starts working with all of the journals that have any source tables journaled to them. *library-name/journal-name* The qualified name of the journal that you want the Capture program to work with. When entering multiple journals, separate the journals with spaces.

Table 55. STRDPRCAP command parameter definitions for System i (continued)

Parameter	Definition and prompts
TRCLMT	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). ***DFT (default)** The Capture program uses the TRACE_LIMIT column value from the IBMSNAP_CAPPARMS table. *trace-limit* The number of minutes of trace data kept in the IBMSNAP_CAPTRACE table after pruning.
MONLMT	Specifies the monitor limit (in minutes). The Capture programs prunes any IBMSNAP_CAPMON table rows that are older than the monitor limit. The default is 10 080 minutes (seven days of monitor entries). ***DFT (default)** The Capture program uses the MONITOR_LIMIT column value from the IBMSNAP_CAPPARMS table. *monitor-limit* The number of minutes of monitor data kept in the IBMSNAP_CAPMON table after pruning.
MONITV	Specifies how frequently (in seconds) the Capture program inserts rows into the IBMSNAP_CAPMON table. The default is 300 seconds (five minutes). ***DFT (default)** The Capture program uses the MONITOR_INTERVAL column value from the IBMSNAP_CAPPARMS table. *monitor-interval* The number of seconds between row insertion into the IBMSNAP_CAPMON table. The monitor interval must be at least 120 seconds (two minutes). If you type a number that is less than 120, this parameter value is set to 120.
MEMLMT	Specifies the maximum size (in megabytes) of memory that the Capture journal job can use. The default is 32 megabytes. ***DFT (default)** The Capture program uses the MEMORY_LIMIT column value from the IBMSNAP_CAPPARMS table. *memory-limit* The maximum number of megabytes for memory.

Table 55. STRDPRCAP command parameter definitions for System i (continued)

Parameter	Definition and prompts
RETAIN	Specifies the new retention limit, which is the number of minutes that data is retained in the change-data (CD), unit-of-work (UOW), IBMSNAP_SIGNAL, and IBMSNAP_AUTHTKN tables before it is removed. This value works with the **CLNUPITV** parameter value. When the **CLNUPITV** value is reached, the CD, UOW, IBMSNAP_SIGNAL, and IBMSNAP_AUTHTKN data is removed if this data is older than the retention limit. Ensure that the Apply intervals are set to copy changed information before the data reaches this **RETAIN** parameter value to prevent inconsistent data in your tables. If the data becomes inconsistent, the Apply program performs a full refresh. The default is 10 080 minutes (seven days). The maximum is 35000000 minutes. **`*DFT (default)`** The Capture program uses the RETENTION_LIMIT column value from the IBMSNAP_CAPPARMS table. *`retention-limit`* The number of minutes that the CD, UOW, IBMSNAP_SIGNAL, and IBMSNAP_AUTHTKN data is retained.
LAG	Specifies the new lag limit, which is the number of minutes that the Capture program can fall behind in processing before restarting. When the lag limit is reached (that is, when the timestamp of the journal entry is older than the current timestamp minus the lag limit), the Capture program initiates a cold start for the tables that it is processing in that journal. The Apply program then performs a full refresh to provide the Capture program with a new starting point. The default is 10 080 minutes (seven days). The maximum is 35000000 minutes. **`*DFT (default)`** The Capture program uses the LAG_LIMIT column value from the IBMSNAP_CAPPARMS table. *`lag-limit`* The number of minutes that the Capture program is allowed to fall behind.

Table 55. STRDPRCAP command parameter definitions for System i (continued)

Parameter	Definition and prompts
`FRCFRQ`	Specifies how often (30 to 600 seconds) that the Capture program writes changes to the change-data (CD) and unit-of-work (UOW) tables. The Capture program makes these changes available to the Apply program either when the buffers are filled or when the FRCFRQ time limit expires, whichever is sooner. Use this parameter to make changes more readily available to the Apply program on servers with a low rate of source table changes. The **FRCFRQ** parameter value is a global value used for all defined source tables. Setting the **FRCFRQ** value to a low number can affect system performance. The default is 30 seconds. **`*DFT (default)`** The Capture program uses the COMMIT_INTERVAL column value from the IBMSNAP_CAPPARMS table. *`force-frequency`* The number of seconds that the Capture program keeps CD and UOW table changes in buffer space before making these changes available to the Apply program.

Usage notes

The **CLNUPITV** parameter on the **`STRDPRCAP`** command specifies the maximum number of hours that the Capture program waits before pruning old records from the change-data (CD), unit-of-work (UOW), 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 **JOBD** parameter uses job queue QDP4/QZSNDPR and the DB2 DataPropagator for System i 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 by 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 System i systems, cold starts work on a journal-by-journal basis. So, for example, if a journal exceeds the lag limit, all replication sources that use the journal are started in cold mode, whereas replication sources that use 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)
```

WRKDPRTRC: Using the DPR trace facility (System i)

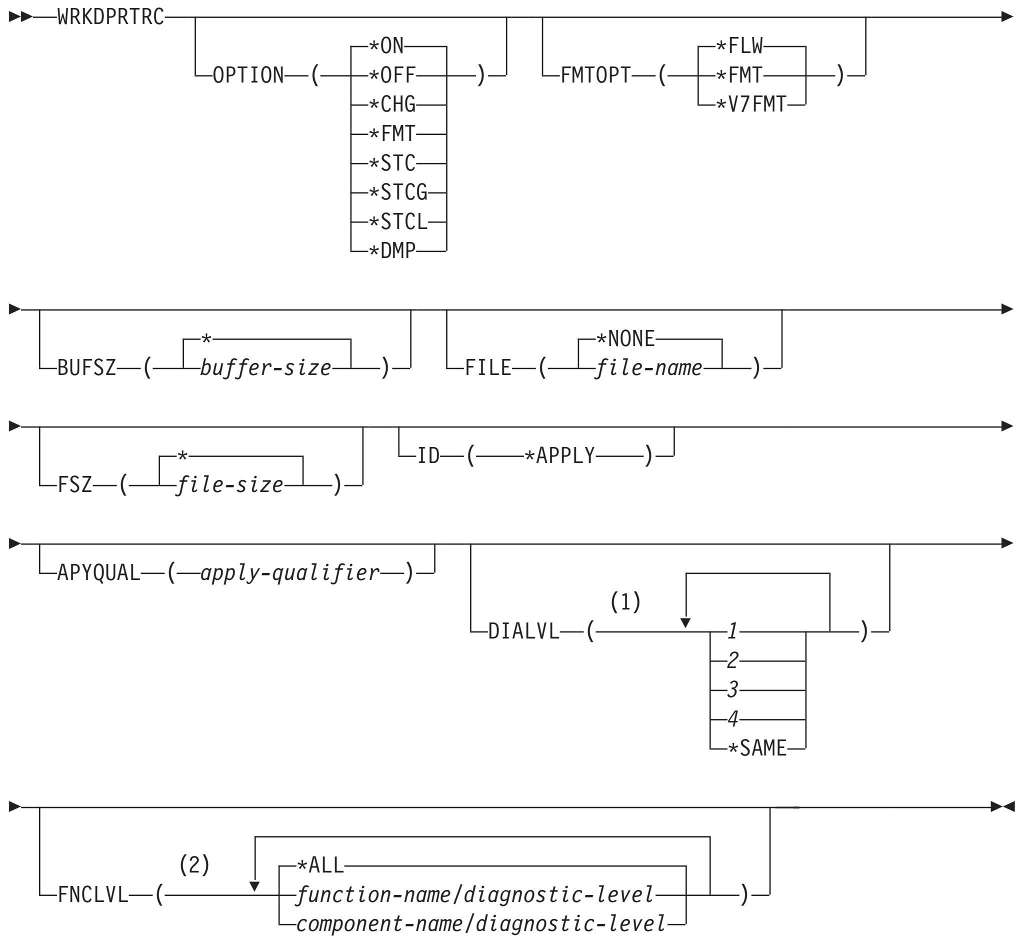
System i

Use the DPR trace (**WRKDPRTRC**) command only if you are instructed to use the command by IBM software support. The command runs the trace facility to log program flow information for specified Apply programs.

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



WRKDPRTRC
OPTION (*ON *OFF *CHG *FMT *STC *STCG *STCL *DMP)
FMTOPT (*FLW *FMT *V7FMT)
BUFSZ (* buffer-size)
FILE (*NONE file-name)
FSZ (* file-size)
ID (*APPLY)
APYQUAL (apply-qualifier)
DIALVL ((1) 1 2 3 4 *SAME)
FNCLVL ((2) *ALL function-name/diagnostic-level component-name/diagnostic-level)

Notes:

1 You can specify multiple values.

2 You can specify up to 20 functions or components.

Table 56 on page 377 lists the invocation parameters.

Table 56. WRKDPRTRC command parameter definitions for System i

Parameter	Definition
OPTION	Specify one trace facility function. ***ON (default)** Turn the trace facility on. This option automatically creates a shared memory segment for tracing. ***OFF** Turn the trace facility off. ***CHG** Change the values of the trace facility parameters. ***FMT** Format the trace facility output from shared memory. ***STC** Display the status of a trace facility. This status information includes the trace version, application version, number of entries, buffer size, amount of buffer used, status code, and program timestamp. This parameter option is equivalent to the stat option of the **asntrc** command used on UNIX, Windows, and z/OS operating systems. ***STCG** Display the status of a trace facility in Replication Center readable format. ***STCL** Display the status of a trace facility with additional version level information. This additional information includes the service levels of each module in the application and appears as long strings of text. This parameter option is equivalent to the statlong option of the **asntrc** command used on UNIX, Windows, and z/OS operating systems. ***DMP** Write the current contents of the trace buffer to a file. When prompting on the **WRKDPRTRC** command, you can press the F4 key to see a list of trace options.
FMTOPT	Specifies the options of the format ID and is used with the **OPTION(*FMT)** parameter. ***FLW (default)** Display the flow of the function calls. ***FMT** Display the format of the trace buffer or trace file. Shows all the detailed data. ***V7FMT** Format the trace buffer or trace file information in Version 7 format. When prompting on the **WRKDPRTRC** command, you can press the F4 key to see a list of format options.

Table 56. WRKDPRTRC command parameter definitions for System i (continued)

Parameter	Definition
BUFSZ	Specifies the size (in bytes) of the trace buffer. You can enter an M, K, or G after the number to indicate megabytes, kilobytes, or gigabytes, respectively. The default is two megabytes. *** (default)** Use the two megabyte default size. *buffer-size* The buffer size in bytes.
FILE	Specifies whether the trace output is written to a file. ***NONE (default)** The trace output goes to shared memory only. *file-name* The name of the output file. If you are using the **OPTION(*DMP)** parameter, this file name represents the name of a dump file.
FSZ	Specifies the size (in bytes) of the file where the trace data is stored. You can enter an M, K, or G after the number to indicate megabytes, kilobytes, or gigabytes, respectively. The default is two gigabytes. *** (default)** Use the two gigabyte default size. *file-size* The file size in bytes.
ID	Specifies the type of program to be traced. ***APPLY (default)** An Apply program trace.
APYQUAL	Specifies the name of Apply program to be traced. *apply-qualifier* The name of the Apply qualifier.

Table 56. WRKDPRTRC command parameter definitions for System i (continued)

Parameter	Definition
`DIALVL`	Specifies the types of trace records to be recorded by the trace facility. Trace records are categorized by a diagnostic mask number: **1** Flow data, which includes the entry and exit points of functions. **2** Basic data, which includes all major events encountered by the trace facility. **3** Detailed data, which includes the major events with descriptions. **4** Performance data. ***SAME** This command uses the diagnostic level settings from the previous trace facility. You can enter one or more diagnostic mask numbers. The numbers that you enter must be in ascending order. Do not type spaces between the numbers. **Important:** The number levels are *not* inclusive. When you start the trace facility, the default is DIALVL(1234). When you subsequently invoke the trace facility, the default is *SAME. When prompting on the **`WRKDPRTRC`** command, you can press the F4 key to see a list of available diagnostic levels.
`FNCLVL`	Specifies if a particular function or component identifier is to be traced. **`*ALL (default)`** All functions and components are included in the trace facility. *`function-name/diagnostic-level`* The name of the function to be traced and the corresponding diagnostic mask numbers. *`component-name/diagnostic-level`* The name of the component to be traced and the corresponding diagnostic mask numbers. You can enter up to 20 function or component names.

Examples for WRKDPRTRC

The following examples illustrate how to use the **`WRKDPRTRC`** command.

Example 1:

To start an Apply trace on the Apply qualifier AQ1 for all functions and components with output written to a file called TRCFILE:

```
WRKDPRTRC OPTION(*ON) FILE(TRCFILE) ID(*APPLY) APYQUAL(AQ1)
```

Example 2:

To end an Apply trace on the Apply qualifier AQ1:

```
WRKDPRTRC OPTION(*OFF) ID(*APPLY) APYQUAL(AQ1)
```

Example 3:

To change an Apply trace on the Apply qualifier AQ1 to diagnostic levels 3 and 4 (detailed and performance data) for all functions and components:

```
WRKDPRTRC OPTION(*CHG) ID(*APPLY) APYQUAL(AQ1) DIALVL(34)
```

Example 4:

To display the status of an Apply trace on the Apply qualifier AQ1:

```
WRKDPRTRC OPTION(*STC) ID(*APPLY) APYQUAL(AQ1)
```

Example 5:

To display the function calls on the Apply qualifier AQ1 at diagnostic levels 3 and 4:

```
WRKDPRTRC OPTION(*FMT) FMTOPT(*FLW) ID(*APPLY) APYQUAL(AQ1) DIALVL (34)
```

Example 6:

To write the Apply trace information of the Apply qualifier AQ1 to a dump file named `DMPFILE`:

```
WRKDPRTRC OPTION(*DMP) FILE(DMPFILE) ID(*APPLY) APYQUAL(AQ1)
```

Chapter 23. 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*.

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 382, Figure 8 on page 383, andFigure 9 on page 384 show the tables at the Capture control server, the columns in each table, and the indexes on each table. Figure 10 on page 385 and Figure 11 on page 386 show the tables at the Apply control server, the columns in each table, and the indexes on each table. Figure 12 on page 387 and Figure 13 on page 388 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)

OS/400 only

***schema*.IBMSNAP_AUTHTKN**
(no unique index)

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

UNIX, Windows, and z/OS only

***schema*.IBMSNAP_CAPENQ**
(no unique index)

LOCKNAME	CHAR(9)

***schema*.IBMSNAP_CAPMON**
(MONITOR_TIME)

MONITOR_TIME	TIMESTAMP NOT NULL
RESTART_TIME	TIMESTAMP NOT NULL
CURRENT_MEMORY	INT NOT NULL
CD_ROWS_INSERTED	INT NOT NULL
RECAP_ROWS_SKIPPED	INT NOT NULL
TRIGR_ROWS_SKIPPED	INT NOT NULL
CHG_ROWS_SKIPPED	INT NOT NULL
TRANS_PROCESSED	INT NOT NULL
TRANS_SPILLED	INT NOT NULL
MAX_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

***schema*.IBMSNAP_CAPPARMS**
(no unique index)

RETENTION_LIMIT	INT
LAG_LIMIT	INT
COMMIT_INTERVAL	INT
PRUNE_INTERVAL	INT
TRACE_LIMIT	INT
MONITOR_LIMIT	INT
MONITOR_INTERVAL	INT
MEMORY_LIMIT	SMALLINT
REMOTE_SRC_SERVER	CHAR(18)
AUTOPRUNE	CHAR(1)
TERM	CHAR(1)
AUTOSTOP	CHAR(1)
LOGREUSE	CHAR(1)
LOGSTDOUT	CHAR(1)
SLEEPINTERVAL	SMALLINT
CAPTURE_PATH	VARCHAR(1040)
STARTMODE	VARCHAR(10)

ASN.IBMSNAP_CAPSCHEMAS
(CAP_SCHEMA_NAME)

CAP_SCHEMA_NAME	VARCHAR(30)

OS/400
ASN.IBMSNAP_CAPSCHEMAS
(CAP_SCHEMA_NAME)

CAP_SCHEMA_NAME	VARCHAR(30)
STATUS	CHAR(1)

***schema*.IBMSNAP_CAPTRACE**
(TRACE_TIME)

OPERATION	CHAR(8) NOT NULL
TRACE_TIME	TIMESTAMP NOT NULL
DESCRIPTION	VARCHAR(1024) NOT NULL

OS/400
***schema*.IBMSNAP_CAPTRACE**
(TRACE_TIME)

OPERATION	CHAR(8) NOT NULL
TRACE_TIME	TIMESTAMP NOT NULL
JOB_NAME	CHAR(26) NOT NULL
JOB_STR_TIME	TIMESTAMP NOT NULL
DESCRIPTION	VARCHAR(298) NOT NULL

***schema*.IBMSNAP_PRUNCNTL**
(SOURCE_OWNER, SOURCE_TABLE, SOURCE_VIEW_QUAL, APPLY_QUAL, SET_NAME, TARGET_SERVER, TARGET_TABLE, TARGET_OWNER, MAP_ID)

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

***schema*.IBMSNAP_PRUNE_LOCK**
(no unique index)

DUMMY	CHAR(1)

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)

***schema*.IBMSNAP_PRUNE_SET**
(TARGET_SERVER, APPLY_QUAL, SET_NAME)

TARGET_SERVER	CHAR(18) NOT NULL
APPLY_QUAL	CHAR(18) NOT NULL
SET_NAME	CHAR(18) NOT NULL
SYNCHTIME	TIMESTAMP
SYNCHPOINT	CHAR(10) FOR BIT DATA NOT NULL

***schema*.IBMSNAP_REGISTER**
(SOURCE_OWNER, SOURCE_TABLE, SOURCE_VIEW_QUAL)

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)

OS/400 only
***schema*.IBMSNAP_REG_EXT**
(VERSION, SOURCE_OWNER, SOURCE_TABLE, SOURCE_VIEW_QUAL)

VERSION	INT NOT NULL
SOURCE _OWNER	VARCHAR(30) NOT NULL
SOURCE_TABLE	VARCHAR(128) NOT NULL
SOURCE_NAME	CHAR(10)
SOURCE_MBR	CHAR(10)
SOURCE_TABLE_RDB	CHAR(18)
JRN_LIB	CHAR(10)
JRN_NAME	CHAR(10)
FR_START_TIME	TIMESTAMP
SOURCE_VIEW_QUAL	SMALLINT NOT NULL
CMT_BEHAVIOR_CASE	SMALLINT NOT NULL WITH DEFAULT
MAX_ROWS_BTWN_CMTS	SMALLINT NOT NULL WITH DEFAULT

***schema*.IBMSNAP_REG_SYNCH**
(TRIGGER_ME)

TRIGGER_ME	CHAR(1) NOT NULL

***schema*.IBMSNAP_RESTART**
(no unique index)

MAX_COMMITSEQ	CHAR(10) FOR BIT DATA NOT NULL
MAX_COMMIT_TIME	TIMESTAMP NOT NULL
MIN_INFLIGHTSEQ	CHAR(10) FOR BIT DATA NOT NULL
CURR_COMMIT_TIME	TIMESTAMP NOT NULL
CAPTURE_FIRST_SEQ	CHAR(10) FOR BIT DATA NOT NULL

OS/400
***schema*.IBMSNAP_RESTART**
(JRN_LIB, JRN_NAME)

MAX_COMMITSEQ	CHAR(10) FOR BIT DATA NOT NULL
MAX_COMMIT_TIME	TIMESTAMP NOT NULL
MIN_INFLIGHTSEQ	CHAR(10) FOR BIT DATA NOT NULL
CURR_COMMIT_TIME	TIMESTAMP NOT NULL
CAPTURE_FIRST_SEQ	CHAR(10) FOR BIT DATA NOT NULL
UID	INTEGER NOT NULL
SEQNBR	BIGINT NOT NULL
JRN_LIB	CHAR(10) NOT NULL
JRN_NAME	CHAR(10) NOT NULL
STATUS	CHAR(1)

***schema*.IBMSNAP_SEQTABLE**
(SEQ)

SEQ	INTEGER NOT NULL

***schema*.IBMSNAP_SIGNAL**
(SIGNAL_TIME)

SIGNAL_TIME	TIMESTAMP NOT NULL WITH DEFAULT
SIGNAL_TYPE	VARCHAR(30) NOT NULL
SIGNAL_SUBTYPE	VARCHAR(30)
SIGNAL_INPUT_IN	VARCHAR(500)
SIGNAL_STATE	CHAR(1) NOT NULL
SIGNAL_LSN	CHAR(10) FOR BIT DATA

***schema*.IBMSNAP_UOW**
(IBMSNAP_COMMITSEQ, IBMSNAP_LOGMARKER)

IBMSNAP_UOWID	CHAR(10) FOR BIT DATA NOT NULL
IBMSNAP_COMMITSEQ	CHAR(10) FOR BIT DATA NOT NULL
IBMSNAP_LOGMARKER	TIMESTAMP NOT NULL
IBMSNAP_AUTHTKN	VARCHAR(30) NOT NULL
IBMSNAP_AUTHID	VARCHAR(30) NOT NULL
IBMSNAP_REJ_CODE	CHAR(1) NOT NULL WITH DEFAULT
IBMSNAP_APPLY_QUAL	CHAR(18) NOT NULL WITH DEFAULT

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**

UNIX, Windows, and z/OS only
(no index)

MAX_COMMITSEQ	CHAR(10) FOR BIT DATA NOT NULL
MAX_COMMIT_TIME	TIMESTAMP NOT NULL
MIN_INFLIGHTSEQ	CHAR(10) FOR BIT DATA NOT NULL
CURR_COMMIT_TIME	TIMESTAMP NOT NULL
CAPTURE_FIRST_SEQ	CHAR(10) FOR BIT DATA NOT NULL

OS/400 only
(JRN_LIB, JRN_NAME)

MAX_COMMITSEQ	CHAR(10) FOR BIT DATA NOT NULL
MAX_COMMIT_TIME	TIMESTAMP NOT NULL
MIN_INFLIGHTSEQ	CHAR(10) FOR BIT DATA NOT NULL
CURR_COMMIT_TIME	TIMESTAMP NOT NULL
CAPTURE_FIRST_SEQ	CHAR(10) FOR BIT DATA NOT NULL
UID	INTEGER NOT NULL
SEQNBR	BIGINT NOT NULL
JRN_LIB	CHAR(10) NOT NULL
JRN_NAME	CHAR(10) NOT NULL
STATUS	CHAR(1)

***schema*.IBMSNAP_SIGNAL**

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**

Informix only
(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 1 of 2)

ASN.IBMSNAP_APPLYTRAIL
(no unique index)

APPLY_QUAL	CHAR(18) NOT NULL
SET_NAME	CHAR(18) NOT NULL
SET_TYPE	CHAR(1) NOT NULL
WHOS_ON_FIRST	CHAR(1) NOT NULL
ASNLOAD	CHAR(1)
FULL_REFRESH	CHAR(1)
EFFECTIVE_MEMBERS	INT
SET_INSERTED	INT NOT NULL
SET_DELETED	INT NOT NULL
SET_UPDATED	INT NOT NULL
SET_REWORKED	INT NOT NULL
SET_REJECTED_TRXS	INT NOT NULL
STATUS	SMALLINT NOT NULL
LASTRUN	TIMESTAMP NOT NULL
LASTSUCCESS	TIMESTAMP
SYNCHPOINT	CHAR(10) FOR BIT DATA
SYNCHTIME	TIMESTAMP
SOURCE_SERVER	CHAR (18) NOT NULL
SOURCE_ALIAS	CHAR(8)
SOURCE_OWNER	VARCHAR(30)
SOURCE_TABLE	VARCHAR(128)
SOURCE_VIEW_QUAL	SMALLINT
TARGET_SERVER	CHAR(18) NOT NULL
TARGET_ALIAS	CHAR(8)
TARGET_OWNER	VARCHAR(30) NOT NULL
TARGET_TABLE	VARCHAR(128) NOT NULL
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)
COMMIT_COUNT	SMALLINT
OPTION_FLAGS	CHAR(4) NOT NULL
EVENT_NAME	CHAR(18)
ENDTIME	TIMESTAMP NOT NULL WITH DEFAULT
SOURCE_CONN_TIME	TIMESTAMP
SQLSTATE	CHAR(5)
SQLCODE	INT
SQLERRP	CHAR(8)
SQLERRM	VARCHAR(70)
APPERRM	VARCHAR(760)

ASN.IBMSNAP_APPENQ
(APPLY_QUAL)

APPLY_QUAL	CHAR(18)

OS/400 only
ASN.IBMSNAP_APPLY_JOB

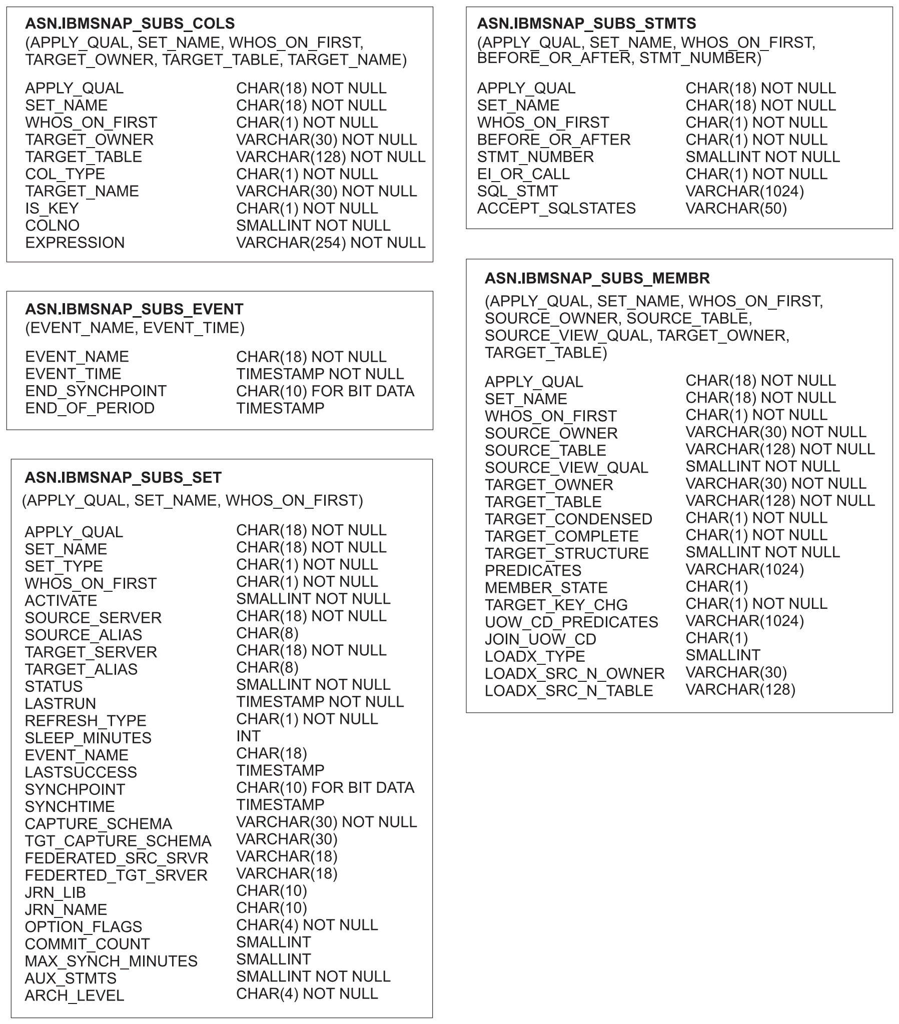
APPLY_QUAL	CHAR(18) NOT NULL
CONTROL_SERVER	CHAR(18) NOT NULL
JOB_NAME	CHAR(10) NOT NULL
USER_NAME	CHAR(10) NOT NULL
JOB_NUMBER	CHAR(6) NOT NULL

ASN.IBMSNAP_APPLYTRACE
(APPLY_QUAL, TRACE_TIME)

APPLY_QUAL	CHAR(18) NOT NULL
TRACE_TIME	TIMESTAMP NOT NULL
OPERATION	CHAR(8) NOT NULL
DESCRIPTION	VARCHAR(1024) NOT NULL

Figure 10. Tables used at the Apply control server. These tables are used by the Apply program at the Apply control server. The columns that make up each table's main index are listed in parentheses under the table name.

Control tables used at the Apply control server (image 2 of 2)



ASN.IBMSNAP_SUBS_COLS
(APPLY_QUAL, SET_NAME, WHOS_ON_FIRST, TARGET_OWNER, TARGET_TABLE, TARGET_NAME)

APPLY_QUAL	CHAR(18) NOT NULL
SET_NAME	CHAR(18) NOT NULL
WHOS_ON_FIRST	CHAR(1) NOT NULL
TARGET_OWNER	VARCHAR(30) 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_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)

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)
FEDERTED_TGT_SRVER	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

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

ASN.IBMSNAP_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

ASN.IBMSNAP_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
PARM_INT	INT
PARM_CHAR	VARCHAR(128)
CONTACT_TYPE	CHAR(1)NOT NULL
CONTACT	VARCHAR(127) NOT NULL

ASN.IBMSNAP_CONTACTGRP
(GROUP_NAME, CONTACT_NAME)

GROUP_NAME	VARCHAR(127) NOT NULL
CONTACT_NAME	VARCHAR(127) NOT NULL

ASN.IBMSNAP_CONTACTS
(CONTACT_NAME)

CONTACT_NAME	VARCHAR(127) NOT NULL
EMAIL_ADDRESS	VARCHAR(128) NOT NULL
ADDRESS_TYPE	CHAR(1) NOT NULL
DELEGATE	VARCHAR(127)
DELEGATE_START	DATE
DELEGATE_END	DATE
DESCRIPTION	VARCHAR(1024)

ASN.IBMSNAP_GROUPS
(GROUP_NAME)

GROUP_NAME	VARCHAR(127) NOT NULL
DESCRIPTION	VARCHAR(1024)

ASN.IBMSNAP_MONENQ
(MONITOR_QUAL)

MONITOR_QUAL	CHAR(18) NOT NULL

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 unique index)

MONITOR_QUAL	CHAR(18) NOT NULL
SERVER_NAME	CHAR(18) NOT NULL
SEVER_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 by using the ASNCLP command-line program or Replication Center. For System i, these control tables are created automatically for you in the ASN library when you install DataPropagator for System i. You can use the System i commands to create Capture control tables in alternate capture schemas.

Table 57 describes the control tables at the Capture server.

Table 57. Quick reference for tables used at the Capture control server

Table name	Description
"IBMSNAP_CAPSCHEMAS table" on page 397	Contains the names of all Capture schemas
IBMSNAP_AUTHTKN table (System i)	Contains information to support update-anywhere replication.
z/OS Linux UNIX Windows "IBMSNAP_CAPENQ table (z/OS, Linux, UNIX, Windows)" on page 391	For each Capture schema, this table is used to ensure that: • Linux UNIX Windows For DB2 for Linux, UNIX and Windows, only one Capture program is running per database. • z/OS For non-data-sharing DB2 for z/OS, only one Capture program is running per subsystem. • z/OS For data-sharing DB2 for z/OS, only one Capture program is running per data-sharing group.

Table 57. Quick reference for tables used at the Capture control server (continued)

Table name	Description
“CD table” on page 400	Contains information about changes that occur at the source. This table is not created until you register a replication source.
“CCD table (non-DB2)” on page 399	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 391	Contains operational statistics that help monitor the progress of the Capture program.
“IBMSNAP_CAPPARMS table” on page 393	Contains parameters that you can specify to control the operations of the Capture program.
“IBMSNAP_CAPTRACE table” on page 398	Contains messages from the Capture program.
“IBMQREP_IGNTRAN table” on page 401	Can be used to inform the Capture program about transactions that you do not want to be captured from the DB2 recovery log.
“IBMQREP_IGNTRANTRC table” on page 402	Records information about transactions that were specified to be ignored.
IBMQREP_PART_HIST table	Maintains a history of changes to partitioned source tables on Linux, UNIX, and Windows systems. This table is used by SQL replication and Q Replication.
“IBMSNAP_PARTITIONINFO table” on page 403	Contains information that enables the Capture program to restart from the earliest required log sequence number.
“IBMSNAP_PRUNE_LOCK table” on page 406	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 407	Coordinates the pruning of CD tables.
“IBMSNAP_PRUNCNTL table” on page 404	Coordinates synch point updates between the Capture and Apply programs.
System i IBMSNAP_REG_EXT (System i)	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 409	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 415	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 416	Contains information that enables the Capture program to resume capturing from the correct point in the log or journal. For System i environments, this table is also used to determine the starting time of the**RCVJRNE** (Receive Journal Entry) command.

Table 57. Quick reference for tables used at the Capture control server (continued)

Table name	Description
"IBMSNAP_SEQTABLE table (Informix)" on page 418	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 418	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 422	Provides additional information about transactions that have been committed to a source table.

IBMSNAP_AUTHTKN table (System i)

The IBMSNAP_AUTHTKN table is used in the System i 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 by using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

Table 58 provides a brief description of the columns in the IBMSNAP_AUTHTKN table.

Table 58. 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 System i 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 System i 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 System i 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 58. 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_CAPENQ table (z/OS, Linux, UNIX, Windows)

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 by using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

The IBMSNAP_CAPENQ table is not used on non-DB2 relational or System i servers.

While running, the Capture program exclusively locks this table.

Table 59 provides a brief description of the column in the IBMSNAP_CAPENQ table.

Table 59. 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 60 provides a brief description of the columns in the IBMSNAP_CAPMON table.

Table 60. 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.
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 (System i)	**Data type**: CHAR(10); **Nullable**: Yes System i The library name of the journal that the Capture program was processing.

Table 60. Columns in the IBMSNAP_CAPMON table (continued)

Column name	Description
JRN_NAME (System i)	**Data type**: CHAR(10); **Nullable**: Yes The name of the journal that the Capture program was processing.
LOGREADLIMIT	**Data type**: INT; **Nullable**: No The number of times that the Capture program paused from reading log records because 1000 records had been read, but no completed transactions had yet been encountered within those 1000 records.
CAPTURE_IDLE	**Data type**: INT; **Nullable**: No The number of times that the Capture program slept because it didn't have any work to process.
SYNCHTIME	**Data type**: TIMESTAMP; **Nullable**: No The current value of SYNCHTIME read from the global row of the register table when the monitor record was inserted into this table.
CURRENT_LOG_TIME	**Data type:** TIMESTAMP; **Nullable:** No The timestamp at the Capture server of the latest database commit that was seen by the Capture log reader.
RESTART_SEQ	**Data type:** VARCHAR(16) FOR BIT DATA; **Nullable:** No, with default The logical log sequence number in the recovery log 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.
CURRENT_SEQ	**Data type:** VARCHAR(16) FOR BIT DATA; **Nullable:** No, with default The most recent logical log sequence number in the recovery log that the Capture program read.
LAST_EOL_TIME	**Data type:** TIMESTAMP; **Nullable:** Yes The time at the Capture server when the Capture program reached the end of the log.
LOGREAD_API_TIME	**Data type:** INTEGER; **Nullable:** Yes The number of milliseconds that the Capture program spent using the DB2 log read application program interface (API) to retrieve log records.
NUM_LOGREAD_CALLS	**Data type:** INTEGER; **Nullable:** Yes The number of log read API calls that the Capture program made.

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 61 provides a brief description of the columns in the IBMSNAP_CAPPARMS table.

Table 61. 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.
PRUNE_INTERVAL	**Data type**: INT; **Nullable**: Yes How often, in seconds, the Capture program automatically prunes (AUTOPRUNE = Y) rows in the CD, UOW, signal, trace, and Capture monitor tables that are no longer needed. A lower prune interval saves space, but increases processing costs. A higher prune interval requires more CD and UOW table space, but decreases processing costs.
TRACE_LIMIT	**Data type**: INT; **Nullable**: Yes The number of minutes that rows remain in the IBMSNAP_CAPTRACE table before they are eligible for pruning. During the pruning process, the rows in the Capture trace table are pruned if the number of minutes (current timestamp - the time a row was inserted in the Capture trace table) exceeds the value of TRACE_LIMIT.
MONITOR_LIMIT	**Data type**: INT; **Nullable**: Yes The number of minutes that rows remain in the IBMSNAP_CAPMON table before they are eligible for pruning. During the pruning process, rows in the Capture monitor table are pruned if the value of minutes (current timestamp - MONITOR_TIME) exceeds the value of MONITOR_LIMIT.
MONITOR_INTERVAL	**Data type**: INT; **Nullable**: Yes How often, in seconds, that the monitor thread adds a row to the Capture monitor IBMSNAP_CAPMON table. For Capture for System i, enter an interval greater than 120 seconds.
MEMORY_LIMIT	**Data type**: SMALLINT; **Nullable**: Yes The amount of memory, in megabytes, that the Capture program is allowed to use. After this allocation is used up, memory transactions will spill to a file.

Table 61. Columns in the IBMSNAP_CAPPARMS table (continued)

Column name	Description
REMOTE_SRC_SERVER	**Data type**: CHAR(18); **Nullable**: Yes Reserved for future options of SQL Replication. Currently this column contains the default value of null.
AUTOPRUNE	**Data type**: CHAR(1); **Nullable**: Yes A flag that indicates whether the Capture program automatically prunes rows that are no longer needed from the CD, UOW, signal, trace, and Capture monitor tables: **Y** Autopruning is on. **N** Autopruning is off.
TERM	**Data type**: CHAR(1); **Nullable**: Yes A flag that indicates whether the Capture program terminates when DB2 is quiesced or stopped: **Y** The Capture program terminates when DB2 is quiesced or stopped. **N** The Capture program stays active and waits for DB2 to be restarted or unquiesced.
AUTOSTOP	**Data type**: CHAR(1); **Nullable**: Yes A flag that indicates whether the Capture program stops capturing changes as soon as it reaches the end of the active logs: **Y** The Capture program stops as soon as it reaches the end of the active logs. **N** The Capture program continues running when it reaches the end of the active logs.
LOGREUSE	**Data type**: CHAR(1); **Nullable**: Yes A flag that indicates whether the Capture program overwrites the Capture log file or appends to it. **Y** The Capture program reuses the log file by first deleting it and then recreating it when the Capture program is restarted. **N** The Capture program appends new information to the Capture log file.
LOGSTDOUT	**Data type**: CHAR(1); **Nullable**: Yes A flag that indicates where the Capture program directs the log file messages: **Y** The Capture program directs log file messages to both the standard out (STDOUT) and the log file. **N** The Capture program directs most log file messages to the log file only. Initialization messages go to both the standard out (STDOUT) and the log file.
z/OS Linux UNIX Windows SLEEP_INTERVAL (z/OS, Linux, UNIX, Windows)	**Data type**: SMALLINT; **Nullable**: Yes The number of seconds that the Capture program sleeps when it reaches the end of the active logs (in Linux, UNIX and Windows, or in z/OS non-data-sharing environments), or when an inefficient amount of data has been returned (in z/OS data-sharing environments).
CAPTURE_PATH	**Data type**: VARCHAR(1040); **Nullable**: Yes The path where the output from the Capture program is sent.

Table 61. Columns in the IBMSNAP_CAPPARMS table (continued)

Column name	Description
STARTMODE	**Data type**: VARCHAR(10); **Nullable**: Yes The processing procedure that the Capture program uses when it is started: **cold** The Capture program deletes all rows in its CD tables and UOW table during initialization. All subscriptions to these replication sources are fully refreshed during the next Apply processing cycle (that is, all data is copied from the source tables to the target tables). If the Capture program tries to cold start but you disabled full refresh, the Capture program will start but the Apply program will fail and will issue an error message. **warmsi** The Capture program warm starts; except if this is the first time you are starting the Capture program then it switches to a cold start. The warmsi start mode ensures that cold starts happen only when you initially start the Capture program. **warmns** The Capture program warm starts. If it can't warm start, it does not switch to cold start. The warmns start mode prevents cold starts from occurring unexpectedly and is useful when problems arise (such as unavailable databases or table spaces) that require repair and that prevent a warm start from proceeding. When the Capture program warm starts, it resumes processing where it ended. If errors occur after the Capture program started, the Capture program terminates and leaves all tables intact.
ARCH_LEVEL	**Data type**: CHAR(4); **Nullable**: No The version of the Capture control tables: **1001** Version 10 on Linux, UNIX, and Windows **100Z** Version 10 on z/OS **0973** Version 9.7 Fix Pack 3 on Linux, UNIX, and Windows **Attention:** When updating the IBMSNAP_CAPPARMS table, do not change the value in this column.
COMPATIBILITY	**Data type**: CHAR(4); **Nullable**: No, with default Determines the length of log sequence numbers in the Capture control tables, CD tables and UOW tables. The 16-byte log sequence numbers are used starting with Version 10 on Linux, UNIX, and Windows. **1001** 16-byte log sequence numbers are used **0801** 10-byte log sequence numbers are used The Apply program uses the value in this column to determine the length of log sequence numbers to use in its control tables and CCD target tables.
LOGRDBUFSZ	**Data type**: INTEGER; **Nullable**: No, with default **Linux, UNIX, and Windows only:** The size in KB of the buffer that the Capture program passes to DB2 when Capture retrieves log records. DB2 fills the buffer with available log records that Capture has not retrieved. Default: 256 KB.

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 by using SQL. Altering this table inappropriately can cause unexpected results.

The following two tables show operating system-specific layouts of the IBMSNAP_CAPSCHEMAS table.

Table 62. Columns in the IBMSNAP_CAPSCHEMAS table for all operating systems other than System i

Column name	Description
CAP_SCHEMA_NAME	**Data type**: VARCHAR(128); **Nullable**: Yes The name of a Capture schema. A row exists for each Capture schema.

Table 63. Columns in the Capture schemas table for System i

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.

IBMQREP_COLVERSION table

The IBMQREP_COLVERSION table is used by the Q Capture and Capture programs to keep track of different versions of a source table.

Server: Q Capture server

Default schema: ASN

Index: LSN, TABLEID1, TABLEID2, POSITION

Index: TABLEID1 ASC, TABLEID2 ASC

Important: Do not alter this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

The Q Capture or Capture program inserts rows into this table when the Q subscription or registration for a source table is first activated, and then each time the source table is altered.

Table 64 provides a brief description of the columns in the IBMQREP_COLVERSION table.

Table 64. Columns in the IBMQREP_COLVERSION table

Column name	Description
LSN	**Data type:** VARCHAR(16) FOR BIT DATA; **Nullable:** No The point in the DB2 recovery log where the Q Capture program or Capture program detected a new version of the source table.
TABLEID1	**Data type:** SMALLINT; **Nullable:** No The object identifier (OBID) in SYSIBM.SYSTABLES.
TABLEID2	**Data type:** SMALLINT; **Nullable:** No The database identifier (DBID) in SYSIBM.SYSTABLES.
POSITION	**Data type:** SMALLINT; **Nullable:** No The ordinal position of the column in the table, starting at 0 for the first column in the table.
NAME	**Data type:** VARCHAR(128); **Nullable:** No The name of the column.
TYPE	**Data type:** SMALLINT; **Nullable:** No An internal data type identifier for the column (SQLTYPE in SYSIBM.SYSCOLUMNS).
LENGTH	**Data type:** INTEGER; **Nullable:** No The maximum data length for this column.
NULLS	**Data type:** CHAR(1); **Nullable:** No A flag that identifies whether the column allows null values: **Y** The column allows null values. **N** The column does not allow null values.
DEFAULT	**Data type:** VARCHAR(1536); **Nullable:** Yes The default value of the column (DEFAULTVALUE) in SYSIBM.SYSCOLUMNS. This column is NULL if there is no default.
CODEPAGE	**Data type:** INTEGER; **Nullable:** Yes The code page that is used for data in this column. The value is 0 if the column is defined as FOR BIT DATA or is not a string type. Default: NULL
SCALE	**Data type:** INTEGER; **Nullable:** Yes The scale of decimal data in decimal columns. The value is 0 for non-decimal columns. Default: NULL

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. z/OS Linux UNIX Windows

Table 65. 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.

System i

Table 66. Columns in the Capture trace table for System i

Column name	Description
OPERATION	**Data type**: CHAR(8); **Nullable**: No The type of operation that the Capture program performed, for example, initialization, capture, or error condition.
TRACE_TIME	**Data type**: TIMESTAMP; **Nullable**: No The time that the row was inserted in the Capture trace table. TRACE_TIME rows that are eligible for trace limit pruning will be deleted when the Capture program prunes the CD and UOW tables.
JOB_NAME	**Data type**: CHAR(26); **Nullable**: No The fully qualified name of the job that wrote this trace entry. **Position** **Description** **1–10** The Capture schema name or the journal job name **11–20** The ID of the user who started the Capture program **21–26** The job number
JOB_STR_TIME	**Data type**: TIMESTAMP; **Nullable**: No The starting time of the job that is named in the JOB_NAME column.
DESCRIPTION	**Data type**: VARCHAR(298); **Nullable**: No The message ID followed by the message text. The message ID is the first seven characters of the DESCRIPTION column. The message text starts at the ninth position of the DESCRIPTION column.

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 by 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. For example, IMS DataPropagator captures 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 67 provides a brief description of the columns in the CCD table.

Table 67. 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 approximate time that the data was committed.
user key columns	If the CCD table is condensed, this column contains the columns that make up the target key.
user non-key columns	The non-key data columns from the source table. The column names that are in the source table do not need to match these column names, but the data types must be compatible.
user computed columns	User-defined columns that are derived from SQL expressions. You can use computed columns with SQL functions to convert source data types to different target data types.

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 by using SQL. Altering this table inappropriately can cause a loss of data.

Table 68 provides a list and a brief description of the columns in the CD table.

Table 68. 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 by using the IBMSNAP_COMMITSEQ column.
IBMSNAP_INTENTSEQ	The log sequence number of the log record of the change (insert, update, or delete). This value is globally ascending. If you selected for updates to be processes as delete/insert pairs, the IBMSNAP_INTENTSEQ value for the delete row is manufactured to be slightly smaller than the corresponding value for the insert row.
IBMSNAP_OPERATION	A flag that indicates the type of operation for a record: **I** Insert **U** Update **D** Delete
user column after-image	In most cases, the after-image column contains the value that is in the source column after the change occurs. This column has the same name, data type, and null attributes as the source column. In the case of an update, this column reflects the new value of the data that was updated. In the case of a delete, this column reflects the value of the data that was deleted. In the case of an insert, this column reflects the value of the data that was inserted.
user column before-image	This column only exists in the CD table if you registered the source to include before-image column values. In most cases, the before-image column contains the value that was in the source column before the change occurred. This column has the same name as the source column, prefixed by the value in the BEFORE_IMG_PREFIX column in the 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.

IBMQREP_IGNTRAN table

The IBMQREP_IGNTRAN table can be used to inform the Q Capture or Capture program about transactions that you do not want to be captured from the DB2 recovery log. You use SQL to insert rows in the table that inform the programs to ignore transactions based on authorization ID, authorization token (z/OS only), or plan name (z/OS only).

Server: Q Capture server, Capture control server

Default schema: ASN

Unique index: AUTHID ASC, AUTHTOKEN ASC, PLANNAME ASC

Table 69 provides a brief description of the columns in the IBMQREP_IGNTRAN table.

Table 69. Columns in the IBMQREP_IGNTRAN table

Column name	Description
AUTHID	**Data type:** CHAR(128); **Nullable:** Yes The primary authorization ID for the transaction that you want to ignore.
AUTHTOKEN	**Data type:** CHAR(30); **Nullable:** Yes z/OS The authorization token (job name) for the transaction that you want to ignore.
PLANNAME	**Data type:** CHAR(8); **Nullable:** Yes z/OS The plan name for the transaction that you want to ignore.
IGNTRANTRC	**Data type:** CHAR(1); **Nullable:** No, with default A flag that tells the Q Capture or Capture program whether to trace transactions that were ignored based on the AUTHID, AUTHTOKEN, or PLANNAME value that was specified in the IBMQREP_IGNTRAN table: **N (default)** Tracing is disabled. **Y** Tracing is enabled. Each time a transaction is ignored, a row is inserted into the IBMQREP_IGNTRANTRC table and a message is issued.

IBMQREP_IGNTRANTRC table

The IBMQREP_IGNTRANTRC table records information about transactions that were specified to be ignored.

Server: Q Capture server, Capture control server

Default schema: ASN

Index: IGNTRAN_TIME ASC

Important: Do not alter this table by using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

A row is inserted in the IBMQREP_IGNTRANTRC table when a transaction is ignored in the DB2 recovery log. This table is pruned according to the `trace_limit` parameter for the Q Capture or Capture program.

Table 70 provides a brief description of the columns in the IBMQREP_IGNTRANTRC table.

Table 70. Columns in the IBMQREP_IGNTRANTRC table

Column name	Description
IGNTRAN_TIME	**Data type:** TIMESTAMP; **Nullable:** No, with default The time when the transaction was ignored. Default: Current timestamp

Table 70. Columns in the IBMQREP_IGNTRANTRC table (continued)

Column name	Description
AUTHID	**Data type:** CHAR(128); **Nullable:** Yes The primary authorization ID of the transaction that was ignored.
AUTHTOKEN	**Data type:** CHAR(30); **Nullable:** Yes z/OS The authorization token (job name) for the transaction that was ignored.
PLANNAME	**Data type:** CHAR(8); **Nullable:** Yes z/OS The plan name for the transaction that was ignored.
TRANSID	**Data type:** CHAR(10) FOR BIT DATA; **Nullable:** No The transaction identifier for the transaction that was ignored.
COMMITLSN	**Data type:** VARCHAR(16) FOR BIT DATA; **Nullable:** No The commit log sequence number or time sequence for the transaction that was ignored.

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 by 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 71 on page 404 provides a brief description of the columns in the IBMSNAP_PARTITIONINFO table.

Table 71. Columns in the IBMSNAP_PARTITIONINFO table

Column name	Description
PARTITIONID	**Data type**: INT; **Nullable**: No The partition ID for each valid partition.
USAGE	**Data type**: CHAR(1); **Nullable**: No The usage of the log sequence number (LSN). An "R" in this column indicates that the LSN has been restarted.
SEQUENCE	**Data type**: CHAR(10) for bit data; **Nullable**: No The restart LSN for the partition 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 partition 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 by 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 72 provides a brief description of the columns in the IBMSNAP_PRUNCNTL table.

Table 72. 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(128); **Nullable**: No The high-level qualifier for the target table or view for this member.

Table 72. Columns in the IBMSNAP_PRUNCNTL table (continued)

Column name	Description
TARGET_TABLE	**Data type**: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: No The name of the target table or view for this member.
SYNCHTIME	**Data type**: TIMESTAMP; **Nullable**: Yes The Capture program sets this timestamp during the initialization handshake process with the Apply program. The value comes from the timestamp of the commit log record that is associated with the transaction of the CAPSTART signal insert. It will not be updated again unless a subsequent initialization process takes place.
SYNCHPOINT	**Data type**: VARCHAR(16) FOR BIT DATA; **Nullable**: Yes The Capture program sets this value during the initialization handshake process with the Apply program. The value comes from the log sequence number of the commit log record that is associated with the transaction of the CAPSTART signal insert. It will not be updated again unless a subsequent initialization process takes place.
SOURCE_OWNER	**Data type**: VARCHAR(128); **Nullable**: No The high-level qualifier of the source table or view for this member.
SOURCE_TABLE	**Data type**: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: No The name of the source table or view for this member.
SOURCE_VIEW_QUAL	**Data type**: SMALLINT; **Nullable**: No This column is used to support multiple registrations for different source views with identical SOURCE_OWNER and SOURCE_TABLE column values. This value is set to 0 for physical tables that are defined as sources and is greater than 0 for views that are defined as sources.
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No The Apply qualifier that identifies which Apply program is processing this member.
SET_NAME	**Data type**: CHAR(18); **Nullable**: No The name of the subscription set that this subscription-set member belongs to.
CNTL_SERVER	**Data type**: CHAR(18); **Nullable**: No The name of the server where the Apply control tables reside for this Apply program, which is identified by the APPLY_QUAL.

Table 72. Columns in the IBMSNAP_PRUNCNTL table (continued)

Column name	Description
TARGET_STRUCTURE	**Data type**: SMALLINT; **Nullable**: No A value that identifies the type of target table or view: **1** Source table **3** CCD table **4** Point-in-time table **5** Base aggregate table **6** Change aggregate table **7** Replica table **8** User copy table **9** CCD table without a join of the IBMSNAP_UOW and CD tables
CNTL_ALIAS	**Data type**: CHAR(8); **Nullable**: Yes The DB2 alias corresponding to the Apply control server named in the CNTL_SERVER column.
PHYS_CHANGE_OWNER	**Data type**: VARCHAR(128); **Nullable**: Yes 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.
PHYS_CHANGE_TABLE	**Data type**: VARCHAR(128); VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier;**Nullable**: Yes The value in the PHYS_CHANGE_TABLE column from the IBMSNAP_REGISTER table that is associated with the source of this particular subscription-set member.
MAP_ID	**Data type**: VARCHAR(10); **Nullable**: No A uniqueness factor that provides a shorter, more easily used index into this table. It is also used to associate CAPSTART inserts into the signal table with the appropriate row in the pruning control table.

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 by 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 by using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

Table 73 provides a brief description of the columns in the IBMSNAP_PRUNE_SET table.

Table 73. 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**: VARCHAR(16) 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 (System i)

The IBMSNAP_REG_EXT table is a System i-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 System i-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 by 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 System i server.

Table 74 provides a brief description of the columns in the IBMSNAP_REG_EXT table.

Table 74. Columns in the IBMSNAP_REG_EXT table

Column name	Description
VERSION	**Data type**: INT; **Nullable**: No The version of DB2 DataPropagator for System i 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 you use 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 74. Columns in the IBMSNAP_REG_EXT table (continued)

Column name	Description
CMT_BEHAVIOR_CASE	**Data type**: SMALLINT; **Nullable**: No, with default; **Default**: 0 An integer that represents how the application programs that are updating the source table use commitment control. The Capture program uses this value to manage its memory usage for CD rows that it has constructed but is not yet ready to write to the CD tables. **-1** The commitment control pattern is not yet established for the applications. This is the initial value in the column. **0** None of the applications that update the source uses commitment control. **1** All of the applications that update the source use commitment control. Therefore, two different applications never update the same source table under commitment control at the same time. **2** For concurrent applications that update the source, some use commitment control and others do not. It is possible that two applications are updating the source table by using commitment control concurrently.
MAX_ROWS_BTWN_CMTS	**Data type**: SMALLINT; **Nullable**: No, with default; **Default**: 0 The maximum number of rows that the Capture program can process before it commits data to the CD table.

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 by 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 75 provides a brief description of the columns in the IBMSNAP_REGISTER table.

Table 75. Columns in the IBMSNAP_REGISTER table

Column name	Description
SOURCE_OWNER	**Data type**: VARCHAR(128); **Nullable**: No The high-level qualifier of the source table or view that you registered.

Table 75. Columns in the IBMSNAP_REGISTER table (continued)

Column name	Description
SOURCE_TABLE	**Data type**: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: No The name of the source table or view that you registered.
SOURCE_VIEW_QUAL	**Data type**: SMALLINT; **Nullable**: No This column is used to support multiple registrations for different source views with identical SOURCE_OWNER and SOURCE_TABLE column values. This value is set to 0 for physical tables that are defined as sources, and is greater than 0 for views that are defined as sources.
GLOBAL_RECORD	**Data type**: CHAR(1); **Nullable**: No
SOURCE_STRUCTURE	**Data type**: SMALLINT; **Nullable**: No A value that identifies the structure of the source table or view: **1** User table **3** CCD table **4** Point-in-time table **5** Base aggregate table **6** Change aggregate table **7** Replica table **8** User copy table **9** CCD table without a join of the IBMSNAP_UOW and CD tables
SOURCE_CONDENSED	**Data type**: CHAR(1); **Nullable**: No A flag that indicates whether the source table is a condensed table, meaning that all rows with the same key are condensed to one row: **Y** The source is condensed. **N** The source is not condensed. **A** The source is a base-aggregate or change-aggregate table.
SOURCE_COMPLETE	**Data type**: CHAR(1); **Nullable**: No A flag that indicates how the source table stores rows of primary key values: **Y** The source table contains a row for every primary key value of interest. **N** The source table contains a subset of rows of primary key values.
CD_OWNER	**Data type**: VARCHAR(128); **Nullable**: Yes The high-level qualifier of the source's CD table. **For tables as sources** For all registered source tables that are not external CCD tables, this column contains the high-level qualifier of the CD table associated with that source table. **For views as sources** This column contains the high-level qualifier of the CD view. **For external CCD tables as sources** This column is null.

Table 75. Columns in the IBMSNAP_REGISTER table (continued)

Column name	Description
CD_TABLE	**Data type**: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: Yes The name of the source's CD table. **For tables as sources** For all registered source tables that are not external CCD tables, this column contains the name of the CD table that holds captured updates of the source table. **For views as sources** This column contains the name of the CD view. **For external CCD tables as sources** This column is null.
PHYS_CHANGE_OWNER	**Data type**: VARCHAR(128); **Nullable**: Yes The high-level qualifier of the table or view that the Apply program uses for change-capture replication: **For tables as sources** For all registered source tables that are not external CCD tables, this column contains the high-level qualifier of the physical CD table that is associated with that source table. **For views as sources** This column contains the high-level qualifier of the physical CD table that is associated with that source view. **For external CCD tables as sources** This column contains the high-level qualifier of the external CCD table or view.
PHYS_CHANGE_TABLE	**Data type**: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: Yes The name of the table or view that the Apply program uses for change-capture replication: **For tables as sources** For all registered source tables that are not external CCD tables, this column contains the name of the physical CD table that is associated with that source table. **For views as sources** This column contains the name of the physical CD table that is associated with that source view. **For external CCDs as sources** This column contains the name of the external CCD table or view.
CD_OLD_SYNCHPOINT	**Data type**: VARCHAR(16) FOR BIT DATA; **Nullable**: Yes This column is used for the initial handshake between the Apply program and the Capture program. The Capture program then begins capturing data from this log sequence number in the source log. This column is also used to show that retention-limit pruning has occurred for a CD table. If this value is null, then the registration is inactive.
CD_NEW_SYNCHPOINT	**Data type**: VARCHAR(16) FOR BIT DATA; **Nullable**: Yes The Capture program advances this column as it inserts new rows into the CD table. The Apply program uses this column to see if there are new changes to be replicated.

Table 75. Columns in the IBMSNAP_REGISTER table (continued)

Column name	Description
DISABLE_REFRESH	**Data type**: SMALLINT; **Nullable**: Yes A flag that indicates whether full refreshes are allowed: **0** Full refreshes are allowed. **1** Full refreshes are prevented.
CCD_OWNER	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 for z/OS Version 8 new-function mode subsystems; **Nullable**: Yes For a source that has an internal CCD table associated with it, this column contains the high-level qualifier of the internal CCD. For an external CCD table, this column is null.
CCD_TABLE	**Data type**: VARCHAR(128); **Nullable**: Yes For a source that has an internal CCD table associated with it, this column contains the name of the internal CCD. For an external CCD table, this column is null.
CCD_OLD_SYNCHPOINT	**Data type**: VARCHAR(16) FOR BIT DATA; **Nullable**: Yes The log sequence number when the CCD table was reinitialized. This column is related to full-refresh processing against CCD tables. The value in this column needs to be changed only when the CCD table is initially or subsequently fully refreshed. This value can be much older than any row remaining in the CCD table. If this column is not maintained, the Apply program that uses the CCD table as a replication source does not know that the CCD table was reinitialized, so it fails to reinitialize complete copies of the CCD source.
SYNCHPOINT	**Data type**: VARCHAR(16) FOR BIT DATA; **Nullable**: Yes In the global row (where GLOBAL_RECORD = Y), the synch point 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 synch point value is advanced by the program that maintains the CCD table to indicate that there is new data available in that CCD table.
SYNCHTIME	**Data type**: TIMESTAMP; **Nullable**: Yes In the global row (where GLOBAL_RECORD = Y), the synchtime represents the timestamp from the last log or journal record processed by the Capture program. If the Capture program has reached the end of the DB2 log, the synchtime is advanced to the current DB2 timestamp. In any row in the IBMSNAP_REGISTER table that contains registration information about a CCD table (internal or external), the synchtime value is advanced by the program that maintains the CCD table to indicate the currency of data available in that CCD table.
CCD_CONDENSED	**Data type**: CHAR(1); **Nullable**: Yes A flag that indicates whether the internal CCD that is associated with this source is condensed, meaning that all rows with the same key are condensed to one row: **Y** The internal CCD is condensed. **N** The internal CCD is not condensed. **NULL** No internal CCD table is defined for this source.

Table 75. Columns in the IBMSNAP_REGISTER table (continued)

Column name	Description
CCD_COMPLETE	**Data type**: CHAR(1); **Nullable**: Yes A flag that indicates whether the internal CCD table that is associated with this source is complete, meaning that it initially contained all the rows from the source table: **N** The internal CCD is not complete. **NULL** No internal CCD table is defined for this source.
ARCH_LEVEL	**Data type**: CHAR(4); **Nullable**: No The architectural level of the replication control tables: **0801** Version 8 SQL Replication **0803** Version 8 SQL Replication with enhanced support for Oracle sources **0805** Version 8 SQL Replication with support for DB2 for z/OS new-function mode
DESCRIPTION	**Data type**: CHAR(254); **Nullable**: Yes A description of the replication source.
BEFORE_IMG_PREFIX	**Data type**: VARCHAR(4); **Nullable**: Yes The one-character prefix that identifies before-image column names in the CD table. The combination of the before-image prefix and the CD column name must be unambiguous, meaning that a prefixed CD column name cannot be the same as a current or potential after-image column name. The length in bytes of the BEFORE_IMG_PREFIX is: **1** For an ASCII or an EBCDIC single byte prefix character. **2** For an ASCII double byte prefix character. **4** For an EBCDIC DBCS prefix character. This length allows for shift-in and shift-out characters.
CONFLICT_LEVEL	**Data type**: CHAR(1); **Nullable**: Yes A flag that indicates the level of conflict detection for this source: **0** The Apply program does not check for conflicts. Data consistency must be enforced by your application to avoid potential conflicting updates. **1** Standard detection with cascading transaction rejection. The Apply program checks for conflicts based on the changes captured to this point. The Apply program will reverse any conflicting transaction at the replica, as well as any transactions with dependencies on the conflicting transaction. Changes captured after the Apply program begins conflict detection will not be checked during this Apply cycle. **2** Enhanced detection with cascading transaction rejection. The Apply program waits until the Capture program captures all changes from the log or journal (see description of the SYNCHTIME column) and then does a standard conflict detection as when set to 1. During the wait time, the Apply program holds a LOCK on the source tables to ensure that no changes are made during the conflict detection process.

Table 75. Columns in the IBMSNAP_REGISTER table (continued)

Column name	Description
CHG_UPD_TO_DEL_INS	**Data type**: CHAR(1); **Nullable**: Yes A flag that indicates how the Capture program stores updates in the CD table. **Y** The Capture program stores updates by 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 by using two rows. This flag ensures that updates made to partitioning columns or columns referenced by a subscription-set predicate are processed correctly. **N** Each update to the source table is stored in a single row in the CD table.
CHGONLY	**Data type**: CHAR(1); **Nullable**: Yes A flag that indicates whether the Capture program captures all changes that occur at the source or only changes that occur in registered columns. Typically you should have this option set to Y to minimize the number of rows that the Capture program inserts into the CD table, but you might want to set this option to N in order to track exactly which rows in the source table were updated. For example, you might just be capturing the primary key column values to audit which rows have been changed in a source table. **Y** The Capture program only captures changes that occur in registered columns in the source table. **N** The Capture program captures changes from all columns in the source table.
RECAPTURE	**Data type**: CHAR(1); **Nullable**: Yes This column is for update-anywhere replication and contains a flag that indicates whether changes that originate from a table or view are recaptured and forwarded to other tables or views. For tables at the master site: **N** Updates to the master that were applied from a replica are not recaptured and will not be replicated to other replicas. **Y** Updates to the master that were applied from a replica and will be replicated to other replicas. For tables at a replica site: **Y** Updates to the replica that were applied from the master are recaptured and are available to be replicated to another table that uses the replica as its source. **N** Updates to the replica that were applied from the master are not recaptured.
OPTION_FLAGS	**Data type**: CHAR(4); **Nullable**: No Reserved for future options of SQL Replication. Currently this column contains the default value of NNNN.

Table 75. Columns in the IBMSNAP_REGISTER table (continued)

Column name	Description
STOP_ON_ERROR	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** Y. A flag that indicates whether the Capture program will terminate or just stop processing the registration if it encounters errors while trying to start, initiate, reinitiate, or insert a row into the CD table: **Y** The Capture program terminates when an error occurs while it is trying to start, initiate, reinitiate, or insert a row into the CD table. **N** The Capture program stops the registration but does not terminate when an error occurs while it is trying to start, reinitialize, or insert a row into the CD table; it continues to process other registrations.
STATE	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** I. A flag that indicates what state the registration is in: **S** The Capture program has stopped processing this registration. The Apply program will not work with this registration until you repair the registration and place it in the I (inactive) state. **A** The registration is active. **I** The registration is inactive.
STATE_INFO	**Data type**: CHAR(8); **Nullable**: Yes; If the Capture program stopped processing the registration, this column contains the error message that was issued regarding the failure.

IBMSNAP_REG_SYNCH table (non-DB2 relational)

The IBMSNAP_REG_SYNCH 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 by using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

Table 76 provides a brief description of the columns in the IBMSNAP_REG_SYNCH table.

Table 76. IBMSNAP_REG_SYNCH table columns

Column name	Description
TRIGGER_ME	**Data type**: CHAR(1); **Nullable**: No A flag of Y that indicates whether a trigger was initiated to update the SYNCHPOINT value for all rows in the register table.

Table 76. 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 by 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.

z/OS Linux UNIX Windows

z/OS, Linux, UNIX, Windows

Table 77. Columns in the IBMSNAP_RESTART table for z/OS, Linux, UNIX, and Windows

Column name	Description
MAX_COMMITSEQ	**Data type**: VARCHAR(16) FOR BIT DATA; **Nullable**: No An internal value that represents the point in the recovery log to which the Capture program has captured and committed to the CD and UOW tables.
MAX_COMMIT_TIME	**Data type**: TIMESTAMP; **Nullable**: No The timestamp that is associated with the value in the MAX_COMMITSEQ column.
MIN_INFLIGHTSEQ	**Data type**: VARCHAR(16) FOR BIT DATA; **Nullable**: No An internal value that represents the point 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 77. Columns in the IBMSNAP_RESTART table for z/OS, Linux, UNIX, and Windows (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**: VARCHAR(16) FOR BIT DATA; **Nullable**: No An internal value that represents the point from which the Capture program started 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.

System i

System i

For System i, 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 78. Columns in the IBMSNAP_RESTART table for System i

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 78. Columns in the IBMSNAP_RESTART table for System i (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 synch point 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 by using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

Table 79 provides a brief description of the column in the IBMSNAP_SEQTABLE table.

Table 79. 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 (synch points) 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

Non-unique 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 80 provides a brief description of the columns in the IBMSNAP_SIGNAL table.

Table 80. Columns in the IBMSNAP_SIGNAL table

Column name	Description
SIGNAL_TIME	**Data type**: TIMESTAMP; **Nullable**: No, with default; **Default**: current timestamp. A timestamp that is used to uniquely identify the row. The Capture program uses this unique value to find the correct row in the signal table to indicate when it has completed processing the Capture signal. This timestamp column is created as NOT NULL WITH DEFAULT, and therefore a Capture signal can generally be inserted in such a way that DB2 supplies the current timestamp as the SIGNAL_TIME value.
SIGNAL_TYPE	**Data type**: VARCHAR(30); **Nullable**: No A flag that indicates the type of signal that was posted: **CMD** A signal posted by you, the Apply program, or another application, which is a well known system command or signal. See the SIGNAL_SUBTYPE column for this table for a list of the available signal subtypes. **USER** A signal posted by you or another user. The Capture program updates the value in the SIGNAL_LSN column with the LSN from the log of when the signal was inserted, and it updates the value in the SIGNAL_STATE column to from P (pending) to R (received).

Table 80. Columns in the IBMSNAP_SIGNAL table (continued)

Column name	Description
SIGNAL_SUBTYPE	**Data type**: VARCHAR(30); **Nullable**: Yes The action that the Capture program performs when a signal from a system command (SIGNAL_TYPE = CMD) occurs. **CAPSTART** The Capture program starts capturing changes at the registered source for a particular subscription-set member, which is identified by the MAP_ID (from the IBMSNAP_PRUNCNTL table) in the SIGNAL_INPUT_IN column. For example, the Apply program issues this signal before it performs a full refresh on all target tables in the set to let the Capture program know that the set is ready to begin change-capture replication. The Apply program posts this signal. **STOP** The Capture program stops capturing changes and terminates. This command can only be issued by you, not the Apply program. **CAPSTOP** The Capture program stops capturing changes for a particular registered source, which is identified by *source_owner.source_table* in the SIGNAL_INPUT_IN column. This command can only be issued by you, not the Apply program. **UPDANY** The Apply program (identified by the Apply qualifier in the SIGNAL_INPUT_IN column) lets the Capture program know that it is working with two Capture programs in an update-anywhere configuration. The Apply program posts this signal. When the signal type is USER, the signal subtype is not used or recognized by the Capture program and therefore is not a required field. It can be set to any value that you want.
SIGNAL_INPUT_IN	**Data type**: VARCHAR(500); **Nullable**: Yes If the SIGNAL_TYPE = USER, then this column contains user-defined input. If the SIGNAL_TYPE = CMD, then the meaning of this value depends on the SIGNAL_SUBTYPE for this signal: **CMD + CAPSTART** The mapping identifier. Because the Capture triggers and not the Capture program process non-DB2 relational sources, there is a trigger called SIGNAL_TRIGGER that fires after the IBMSNAP_SIGNAL table is updated, which updates the IBMSNAP_PRUNCNTL table with the next value in the sequence. **CMD + UPDANY** The Apply qualifier that identifies the Apply program in the update-anywhere configuration. **CMD + CAPSTOP** The name of the source owner and source table that the Capture program should stop capturing changes for (*source_owner.source_table*).

Table 80. Columns in the IBMSNAP_SIGNAL table (continued)

Column name	Description
SIGNAL_STATE	**Data type**: CHAR(1); **Nullable**: No A flag that indicates the status of the signal: **P** The signal is pending; the Capture program has not received it yet. When you post a signal, set the SIGNAL_STATE to P. **R** The Capture program has received the signal. The Capture program sets the SIGNAL_STATE set to R (instead of changing it to C for complete) when it receives a signal where SIGNAL_TYPE = USER, or one where SIGNAL_TYPE = CMD and SIGNAL_SUBTYPE = STOP. **C** The Capture program has completed processing the signal. The Capture program sets this value to C when SIGNAL_TYPE = CMD for all SIGNAL_SUBTYPE values except STOP.
SIGNAL_LSN	**Data type**: VARCHAR(16) FOR BIT DATA; **Nullable**: Yes The log sequence number of the commit record. This value is set only by the Capture program.

System i On System i, 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.

IBMQREP_TABVERSION table

The IBMQREP_TABVERSION table is used by the Q Capture and Capture programs to keep track of different versions of a source table. The Q Capture or Capture program inserts rows into this table when the Q subscription or registration for a source table is first activated, and then each time the source table is altered.

Server: Q Capture server

Default schema: ASN

Index: LSN, TABLEID1, TABLEID2, VERSION

Index: SOURCE_OWNER ASC, SOURCE_NAME ASC

Index: TABLEID1 ASC, TABLEID2 ASC

Important: Do not alter this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

Table 81 on page 422 provides a brief description of the columns in the IBMQREP_TABVERSION table.

Table 81. Columns in the IBMQREP_TABVERSION table

Column name	Description
LSN	**Data type:** VARCHAR(16) FOR BIT DATA; **Nullable:** No The point in the DB2 recovery log where the Q Capture program or Capture program detected a new version of the source table.
TABLEID1	**Data type:** SMALLINT; **Nullable:** No The database identifier (DBID) in SYSIBM.SYSTABLES.
TABLEID2	**Data type:** SMALLINT; **Nullable:** No The object identifier (OBID) in SYSIBM.SYSTABLES.
VERSION	**Data type:** INTEGER; **Nullable:** No A number generated by the Q Capture or Capture program to keep track of the different versions of a source table.
SOURCE_OWNER	**Data type:** VARCHAR(128); **Nullable:** No The schema or high-level qualifier of the source table.
SOURCE_NAME	**Data type:** VARCHAR(128); **Nullable:** No The name of the source table.

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 and type 9 CCD, 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 by using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

System i

- Because Capture for System i 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 System i 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**: VARCHAR(16) 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**: VARCHAR(16) 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 approximate 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 i5/OS, 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(128); **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 i5/OS, 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	**Data type**: CHAR(1); **Nullable**: No, with default; **Default**: 0. A flag that indicates whether any rows were rejected and rolled back. This value is set only during update-anywhere replication if conflict detection is specified as standard or enhanced when you defined your replication source. You can select that this column be part of a noncomplete CCD target table. **0** No known conflicts occurred in the transaction. **1** A conflict occurred because the same row in the master and replica was updated. The value of WHOS_ON_FIRST in the Apply control tables is F. The transaction at the replica was rejected and rolled back. **2** The transaction was rejected and rolled back because it was dependent on a prior transaction that was rejected. The value of WHOS_ON_FIRST in the Apply control tables is F. The prior transaction was rejected because the same row in the master and replica was updated, and the transaction at the replica was rejected and rolled back. **3** The transaction was rejected and rolled back because it contained at least one referential-integrity constraint violation. Because this transaction violates the referential constraints defined on the source table, the Apply program will mark this subscription set as failed. Updates cannot be copied until you correct the referential integrity definitions. **4** The transaction was rejected and rolled back because it was dependent on a prior transaction that was rejected. The prior transaction was rejected because it contained at least one referential-integrity constraint violation. **5** A conflict occurred because the same row in the master and replica was updated. The value of WHOS_ON_FIRST in the Apply control tables is S. The transaction at the replica was rejected and rolled back. **6** The transaction was rejected and rolled back because it was dependent on a prior transaction that was rejected. The value of WHOS_ON_FIRST in the Apply control tables is S. 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.
IBMSNAP_APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No, with default; **Default**: current user name. The Apply qualifier that identifies which Apply program applied the changes. You can select that this column be part of a noncomplete CCD target table.

Tables at the Apply control server

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 by using the ASNCLP command-line program or Replication Center. For System i, these control tables are created automatically for you when install DataPropagator for System i.

Table 83 on page 425 describes the control tables at the Apply server.

Table 83. Control tables at the Apply server

Table name	Description
"ASN.IBMSNAP_APPENQ table"	Used to ensure that only one Apply program is running per Apply qualifier.
"ASN.IBMSNAP_APPLEVEL table" on page 426	Stores the version of the Apply control tables.
System i ASN.IBMSNAP_APPLY_JOB table (System i)	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_APPLYMON table" on page 427	Contains information on the status of the Apply program.
"ASN.IBMSNAP_APPLYTRACE table" on page 432	Contains important messages from the Apply program.
"ASN.IBMSNAP_APPLYTRAIL table" on page 432	Contains audit-trail information about the Apply program.
"ASN.IBMSNAP_APPPARMS table" on page 428	Contains parameters that you can modify to control the operations of the Apply program.
"ASN.IBMSNAP_FEEDETL table" on page 438	Identifies SQL Replication subscription-set members that are used by InfoSphere DataStage to feed a data warehouse.
"ASN.IBMSNAP_SUBS_COLS table" on page 439	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 440	Contains events that you define to control when the Apply program processes a subscription set.
"ASN.IBMSNAP_SUBS_MEMBR table" on page 441	Identifies a source and target table pair and specifies processing information for that pair.
"ASN.IBMSNAP_SUBS_SET table" on page 445	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 450	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 on System i.

Server: Apply control server

Index: APPLY_QUAL

Important: Use caution when you update this table by using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

Table 84 on page 426 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_APPLEVEL table

The IBMSNAP_APPLEVEL table stores the version of the Apply control tables. The value that is stored here is used with the value of the ARCH_LEVEL and COMPATIBILITY columns in the IBMSNAP_CAPPARMS table to determine the length of log sequence numbers that are used in both Capture and Apply control tables.

Server: Apply control server

Table 85 provides a brief description of the column in the IBMSNAP_APPLEVEL table.

Table 85. Column in the IBMSNAP_APPLEVEL table

Column name	Description
ARCH_LEVEL	**Data type**: CHAR(4); **Nullable**: No, with default The version of the Apply control tables. For Version 10.1 on Linux, UNIX, and Windows, the default is 1001.

ASN.IBMSNAP_APPLY_JOB (System i)

The IBMSNAP_APPLY_JOB table, which is System i-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 by using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

Table 86 provides a brief description of the columns in the IBMSNAP_APPLY_JOB table.

Table 86. Columns in the IBMSNAP_APPLY_JOB table

Column name	Description
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No A unique identifier for a group of subscription sets. This value is supplied by the user when defining a subscription set. Each instance of the Apply program is started with an APPLY_QUAL value. This value is used during update-anywhere replication to prevent circular replication of the changes made by the Apply program.

Table 86. Columns in the IBMSNAP_APPLY_JOB table (continued)

Column name	Description
CONTROL_SERVER	**Data type**: CHAR(18); **Nullable**: No The name of the database where the Apply control tables and view are defined.
JOB_NAME	**Data type**: CHAR(10); **Nullable**: No The fully qualified name of the job that wrote this trace entry: **Position 1–10** APPLY_QUAL **Position 11-20** The ID of the user who started the Apply program **Position 21-26** The job number
USER_NAME	**Data type**: CHAR(10); **Nullable**: No The name of the user who started a new instance of the Apply program.
JOB_NUMBER	**Data type**: CHAR(6); **Nullable**: No The job number of the current job for a particular journal. If the journal is not active, this column contains the job number of the last job that was processed.

ASN.IBMSNAP_APPLYMON table

The IBMSNAP_APPLYMON table contains information about the status of the Apply program, including which subscription sets and subscription-set members Apply is currently processing.

Server: Apply control server

Non-unique index: MONITOR_TIME, APPLY_QUAL, WHOS_ON_FIRST

Do not alter this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

Table 87 provides a brief description of the columns in the IBMSNAP_APPLYMON table.

Table 87. Columns in the IBMSNAP_APPLYMON table

Column name	Description
MONITOR_TIME	**Data type**: TIMESTAMP; **Nullable**: No The timestamp at the Apply control server when the most current row of Apply status information was inserted.
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No The Apply qualifier to which this row of Apply status information pertains.

Table 87. Columns in the IBMSNAP_APPLYMON table (continued)

Column name	Description
WHOS_ON_FIRST	**Data type**: CHAR(1); **Nullable**: Yes An indicator of the type of subscription that the Apply program was processing. **F** Update anywhere. The source table is the replica and the target is the master table. **S** The source table is the master and the target is a table other than master. Could be an update-anywhere or read-only subscription.
STATE	**Data type**: SMALLINT; **Nullable**: Yes The state of the Apply program: **0** Fetching from source **1** Applying to target **2** Sleeping **3** Reading control tables **4** Updating control tables
CURRENT_SETNAME	**Data type**: CHAR(18); **Nullable**: Yes The set that Apply was processing. This column contains a null value if Apply is not processing a subscription set.
CURRENT_TABOWNER	**Data type**: VARCHAR(128); **Nullable**: Yes The owner of the target or source table that Apply is working with. This column contains a null value if Apply is reading or updating control table, sleeping, or applying to target tables in transactional order.
CURRENT_TABNAME	**Data type**: VARCHAR(128); **Nullable**: Yes The target or source table that Apply is working with. This column contains a null value if Apply is reading or updating a control table, sleeping, or applying to target tables in transactional order.

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 88 on page 429 provides a brief description of the columns in the IBMSNAP_APPPARMS table.

Table 88. Columns in the IBMSNAP_APPPARMS table

Column name	Description
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No The Apply qualifier matches the parameters to the Apply program to which these parameters apply.
APPLY_PATH	**Data type**: VARCHAR(1040); **Nullable**: Yes The location of the work files used by the Apply program. The default is the directory where the program was started.
CAF	**Data type**: CHAR(1); **Nullable**: Yes **Default:** Y A flag that specifies whether the Apply program uses Call Attach Facility (CAF) connect. **Y (default)** The Apply program overrides the Recoverable Resource Manager Services (RRS) connect and runs with CAF connect. **N** The Apply program uses Recoverable Resource Manager Services (RRS) connect.
COPYONCE	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** N A flag that indicates whether the Apply program executes one copy cycle for each subscription set that is eligible at the time the Apply program is invoked. **Y** The Apply program executes one copy cycle for each eligible subscription set. **N** The Apply program does not execute one copy cycle for each eligible subscription set.
DELAY	**Data type**: INT; **Nullable**: Yes, with default; **Default:** 6 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.
ERRWAIT	**Data type**: INT; **Nullable**: Yes, with default; **Default:** 300 The number of seconds (1 to 300) that the Apply program waits before retrying after the program encounters an error condition. This parameter is ignored if **`copyonce`** is specified.
INAMSG	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** Y A flag that indicates whether the Apply program issues a message when it is inactive. **Y** The Apply program issues a message when inactive. **N** The Apply program does not issue a message when inactive.
LOADXIT	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** N A flag that indicates whether the Apply program invokes the IBM-supplied exit routine (ASNLOAD) that uses the export and load utilities to refresh target tables. **Y** The Apply program invokes ASNLOAD. **N** The Apply program does not invoke ASNLOAD.

Table 88. Columns in the IBMSNAP_APPPARMS table (continued)

Column name	Description
LOGREUSE	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** N A flag that indicates whether the Apply program overwrites the Apply log file or appends to it. **Y** The Apply program reuses the log file by first deleting it and then recreating it when the Apply program is restarted. **N** The Apply program appends new information to the Apply log file.
LOGSTDOUT	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** N A flag that indicates where the Apply program directs the log file messages: **Y** The Apply program directs log file messages to both the standard out (STDOUT) and the log file. **N** The Apply program directs most log file messages to the log file only. Initialization messages go to both the standard out (STDOUT) and the log file.
MONITOR_ENABLED	**Data type**: CHAR(1); **Nullable**: No A flag that indicates whether the Apply program makes inserts into the IBMSNAP_APPLYMON table to record its status: **Y** The Apply program makes inserts into IBMSNAP_APPLYMON on a schedule that is based on the MONITOR_INTERVAL column value. **N (default)** The Apply program does not make inserts into IBMSNAP_APPLYMON.
MONITOR_INTERVAL	**Data type**: INTEGER; **Nullable**: No, with default; **Default:** 60000 milliseconds (1 minute) How often, in milliseconds, the Apply program adds a row to the IBMSNAP_APPLYMON table.
NOTIFY	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** N A flag that indicates whether the Apply program should invoke the exit routine (ASNDONE) that returns control to you after the Apply program finishes copying a subscription set. **Y** The Apply program invokes ASNDONE. **N** The Apply program does not invoke ASNDONE.
OPT4ONE	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** N A flag that indicates whether the performance of the Apply program is optimized if only one subscription set is defined for the Apply program. **Y** The performance of the Apply program is optimized for one subscription set. **N** The performance of the Apply program is not optimized for one subscription set. This parameter is ignored if `copyonce` is specified.

Table 88. Columns in the IBMSNAP_APPPARMS table (continued)

Column name	Description
SLEEP	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** Y A flag that indicates how the Apply program is to proceed if no new subscription sets are eligible for processing: **Y** The Apply program goes to sleep. **N** The Apply program stops. This parameter is ignored if **`copyonce`** is specified.
SQLERRCONTINUE	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** N A flag that indicates whether the Apply program continues processing after it checks the SQLSTATE file for errors. **Y** The Apply program checks the SQLSTATE file for any SQL errors during processing. If an error is found, Apply stops processing. **N** The Apply program does not check the SQLSTATE file and continues processing.
SPILLFILE	**Data type**: VARCHAR(10); **Nullable**: Yes, with default. A flag that indicates where the fetched answer set is stored. z/OS Valid values are: **mem (default)** A memory file. If there is insufficient memory for the answer set, the Apply program uses a disk file. **disk** A disk file. Linux UNIX Windows Valid values are: **disk (default)** A disk file.
TERM	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** Y A flag that indicates whether the Apply program terminates if it cannot connect to its control server. **Y (default)** By default, the Apply program terminates if it cannot connect to its control server. **N** The Apply program does not terminate. Instead, Apply logs an error, waits for the amount of time set by the **`errwait`** parameter, then retries the connection. This parameter is ignored if **`copyonce`** is specified.
TRLREUSE	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** N A flag that indicates whether the Apply program invokes the IBM-supplied exit routine (ASNLOAD) that uses the export and load utilities to refresh target tables: **Y** The Apply program invokes ASNLOAD. **y** The Apply program does not invoke ASNLOAD.

Table 88. Columns in the IBMSNAP_APPPARMS table (continued)

Column name	Description
REFRESH_COMMIT_CNT	**Data type**: INTEGER; **Nullable**: Yes, with default; **Default:** null Specifies the number of rows that the Apply program inserts into the target table before it issues a COMMIT statement. Values can range from 0 to 134217727. A default value of null or 0 means that only one commit is issued after all rows have been inserted; no intermediate commits are issued. This option is not supported for CCD sources.

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 89 provides a brief description of the column in the IBMSNAP_APPLYTRACE table.

Table 89. Columns in the IBMSNAP_APPLYTRACE table

Column name	Description
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No Uniquely identifies which Apply program inserted the message.
TRACE_TIME	**Data type**: TIMESTAMP; **Nullable**: No The time at the Apply control server when the row was inserted into this table.
OPERATION	**Data type**: CHAR(8); **Nullable**: No The type of Apply program operation, for example, initialization, apply, or error condition.
DESCRIPTION	**Data type**: VARCHAR(1024); **Nullable**: No The message ID followed by the message text. The message ID is the first seven characters of the DESCRIPTION column. The message text starts at the ninth position of the DESCRIPTION column.

ASN.IBMSNAP_APPLYTRAIL 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 90 provides a brief description of the columns in the IBMSNAP_APPLYTRAIL table.

Table 90. 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. **F** (first) The source table is the replica and the target table is the master. In the case of update conflicts between the replica and the master table, the replica will have its conflicting transactions rejected. F is not used for read-only subscriptions; it is used for update anywhere. **S** (second) The source table is the master table or other source, and the target table is the replica or other copy. In the case of update conflicts between the master and the replica table, the replica will have its conflicting transactions rejected. S is used for all read-only subscriptions.
ASNLOAD	**Data type**: CHAR(1); **Nullable**: Yes The value used to start the Apply program: **Y** Indicates that the Apply program was started with the parameter `loadxit=y` causing the ASNLOAD user exit routine to be called to perform a full refresh on a subscription set. **N** Indicates that the ASNLOAD exit routine was not called because either a full refresh was not needed or the Apply program was not started with the `loadxit` parameter. **NULL** Indicates that an Apply program error occurred before the Apply program could determine whether the ASNLOAD exit routine should be called.
FULL_REFRESH	**Data type**: CHAR(1); **Nullable**: Yes A flag that indicates whether a full refresh occurred: **Y** Indicates that a full refresh was done for a subscription set. **N** Indicates that a full refresh was not done for a subscription set. **NULL** Indicates that an error occurred before the Apply program could determine whether or not a full refresh was needed.
EFFECTIVE_MEMBERS	**Data type**: INT; **Nullable**: Yes The number of subscription-set members that are changed during an Apply cycle, either by a full refresh or by the replication of inserts, updates, and deletes. This number ranges between zero and the number of defined subscription-set members.

Table 90. Columns in the IBMSNAP_APPLYTRAIL table (continued)

Column name	Description
SET_INSERTED	**Data type**: INT; **Nullable**: No The total number of rows inserted into subscription-set members during the subscription cycle.
SET_DELETED	**Data type**: INT; **Nullable**: No The total number of rows deleted from subscription-set members during the subscription cycle.
SET_UPDATED	**Data type**: INT; **Nullable**: No The total number of rows updated in subscription-set members during the subscription cycle.
SET_REWORKED	**Data type**: INT; **Nullable**: No The total number of rows that the Apply program reworked during the last cycle. The Apply program reworks changes under the following conditions: • If an insert fails because the row already exists in the target table, the Apply program converts the insert to an update of the existing row. • If the update fails because the row does not exist in the target table, the Apply program converts the update to an insert.
SET_REJECTED_TRXS	**Data type**: INT; **Nullable**: No The total number of transactions that were rejected due to an update-anywhere conflict. This column is used only for update-anywhere subscription sets where conflict detection is defined as standard or advanced.

Table 90. Columns in the IBMSNAP_APPLYTRAIL table (continued)

Column name	Description
STATUS	**Data type**: SMALLINT; **Nullable**: No A value that represents the work status for the Apply program after a given cycle: **-1** The replication failed. The Apply program backed out the entire set of rows that it had applied, and no data was committed. If the startup parameter SQLERRCONTINUE = Y, the SQLSTATE that is returned to the Apply program during the last cycle is *not* one of the acceptable errors you indicated in the input file for SQLERRCONTINUE (*apply qualifier*.SQS). **0** The Apply program processed the subscription set successfully. If the startup parameter SQLERRCONTINUE = Y, the Apply program did not encounter any SQL errors that you indicated for the SQLERRCONTINUE startup parameter (in *apply_qualifier*.SQS) and did not reject any rows. **2** The Apply program is processing the subscription set in multiple cycles. It successfully processed a single logical subscription that was divided according to the MAX_SYNCH_MINUTES control column. **16** The Apply program processed the subscription set successfully and returned a status of 0; however, it encountered some SQL errors that you indicated for the SQLERRCONTINUE startup parameter (in *apply_qualifier*.SQS) and rejected some of the rows. See the *apply_qualifier*.ERR file for details about the rows that failed. **Example**: You set SQLERRCONTINUE = Y and indicate that the allowable SQL state is 23502 (SQL code -407). A 23502 error occurs, but no other errors occur. The Apply program finishes processing the subscription set, and it sets the status to 16. On the next execution, a 23502 error occurs, but then a 07006 (SQL code -301) occurs. Now the Apply program stops processing the subscription set, backs out the entire set of rows it had applied, and sets the status to -1 (because no data was committed). **18** The Apply program is processing the subscription set in multiple cycles and returned a status of 2, which means that it successfully processed a single logical subscription that was divided according to the MAX_SYNCH_MINUTES control column. However, it encountered some SQL errors that you indicated for the SQLERRCONTINUE startup parameter (in *apply_qualifier*.SQS) and rejected some of the rows. See the *apply_qualifier*.ERR file for details about the rows that failed.
LASTRUN	**Data type**: TIMESTAMP; **Nullable**: No The estimated time that the last subscription began. The Apply program sets the LASTRUN value each time a subscription set is processed. It is the approximate time at the Apply control server that the Apply program begins processing the subscription set.
LASTSUCCESS	**Data type**: TIMESTAMP; **Nullable**: Yes The Apply control server timestamp for the beginning of the last successful processing of a subscription set.
SYNCHPOINT	**Data type**: VARCHAR(16) FOR BIT DATA; **Nullable**: Yes The Apply program uses this column to record its progress, indicating that it has processed data up to this synch point value for the subscription set.

Table 90. 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(128); **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(128); **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(128); **Nullable**: No The schema name of the Capture server tables for this subscription set.
TGT_CAPTURE_SCHEMA	**Data type**: VARCHAR(128); **Nullable**: Yes If the target table is also the source for another subscription set (such as an external CCD table in a multi-tier configuration or a replica table in an update-anywhere configuration), this column contains the Capture schema that will be used when the table is acting as a source.
FEDERATED_SRC_SRVR	**Data type**: VARCHAR(18); **Nullable**: Yes The name of the federated remote server that is the source for the subscription set, which applies only to non-DB2 relational sources.

Table 90. Columns in the IBMSNAP_APPLYTRAIL table (continued)

Column name	Description
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 System i This column, which applies only to System i Capture servers, is the library name of the journal that the source table uses.
JRN_NAME	**Data type**: CHAR(10); **Nullable**: Yes System i This column, which applies only to System i 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.
APPERRM	**Data type**: VARCHAR(760); **Nullable**: Yes The Apply error message ID and text for a failed execution.

ASN.IBMSNAP_FEEDETL table

The ASN.IBMSNAP_FEEDETL table identifies SQL Replication subscription-set members that are used by InfoSphere DataStage to feed a data warehouse. The table also stores DataStage-related synchpoint information for the subscription-set members that is used to track DataStage progress.

Server: Apply control server

Primary key: APPLY_QUAL, SET_NAME, SOURCE_OWNER, SOURCE_TABLE, TARGET_OWNER, TARGET_TABLE

Important: Use caution when you update this table with 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_FEEDETL table.

Table 91. Columns in the IBMSNAP_FEEDETL 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 to which this member belongs.
SOURCE_OWNER	**Data type**: VARCHAR(128); **Nullable**: No The high-level qualifier for the source table or view for this member.
SOURCE_TABLE	**Data type**: VARCHAR(128); **Nullable**: No The name of the source table or view for this member.
TARGET_OWNER	**Data type**: VARCHAR(128); **Nullable**: No The high-level qualifier for the target table or view for this member.
TARGET_TABLE	**Data type**: VARCHAR(128); **Nullable**: No The name of the target table or view for this member.
MIN_SYNCHPOINT	**Data type**: CHAR(10) FOR BIT DATA; **Nullable**: No The start point for a range of rows that are extracted from the CCD table member and fed to DataStage. The first time the extract job runs, the value is x'00000000000000000000'. At end of a successful DataStage cycle, this column is updated with the MAX_SYNCHPOINT value for the row.
MAX_SYNCHPOINT	**Data type**: CHAR(10) FOR BIT DATA; **Nullable**: No The end point for a range of rows that are extracted from the CCD table member and fed to DataStage. At the start of the cycle, this column is updated with the value in the SYNCHPOINT column of the IBMSNAP_SUBSET table for the subscription set that contains the CCD table member. At the end of the cycle, this value is inserted into the MIN_SYNCHPOINT column and becomes the start point for the next DataStage cycle.
DSX_CREATE_TIME	**Data type**: TIMESTAMP; **Nullable**: No The time when the new DataStage definition (.dsx) file is created.

Table 91. Columns in the IBMSNAP_FEEDETL table (continued)

Column name	Description
DSX_UPDATE_TIME	**Data type**: TIMESTAMP; **Nullable**: No The time when the DataStage definition (.dsx) file is updated.

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 by 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_COLS table.

Table 92. 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. **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(128); **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 92. Columns in the IBMSNAP_SUBS_COLS table (continued)

Column name	Description
COL_TYPE	**Data type**: CHAR(1); **Nullable**: No A flag that indicates the type of column: **A** An after-image column. **B** A before-image column. **C** A computed column or an SQL expression that uses scalar functions. **F** A computed column that uses column functions. **L** A LOB indicator value. **P** A before-image predicate column. **R** A relative record number column, provided by the system and used as a primary key column. Used only by DB2 DataPropagator for System i.
TARGET_NAME	**Data type**: VARCHAR(128); **Nullable**: No The name of the target table or view column. It does not need to match the source column name. Internal-CCD column names cannot be renamed. They must match the source-table column names.
IS_KEY	**Data type**: CHAR(1); **Nullable**: No A flag that indicates whether the column is part of the target key, which can be either a unique index or primary key of a condensed target table: **Y** The column is all or part of the target key. **N** The column is not part of the target key.
COLNO	**Data type**: SMALLINT; **Nullable**: No The numeric location of the column in the original source, to be preserved relative to other user columns in displays and subscriptions.
EXPRESSION	**Data type**: VARCHAR(1024); **Nullable**: No The source column name or an SQL expression used to create the target column contents.

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 by using SQL.

You insert a row into this table when you create a new event to start an Apply program.

Table 93 on page 441 provides a brief description of the columns in the IBMSNAP_SUBS_EVENT table.

Table 93. 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**: VARCHAR(16) 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 94 provides a brief description of the columns in the IBMSNAP_SUBS_MEMBR table.

Table 94. 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 94. Columns in the IBMSNAP_SUBS_MEMBR table (continued)

Column name	Description
WHOS_ON_FIRST	**Data type**: CHAR(1); **Nullable**: No The following values are used to control the order of processing in update-anywhere replication scenarios. **F** (first) The source table is the replica and the target table is the master. In the case of update conflicts between the replica and the master table, the replica will have its conflicting transactions rejected. F is not used for read-only subscriptions; it is used for update anywhere. **S** (second) The source table is the master table or other source, and the target table is the replica or other copy. In the case of update conflicts between the master and the replica table, the replica will have its conflicting transactions rejected. S is used for all read-only subscriptions.
SOURCE_OWNER	**Data type**: VARCHAR(128); **Nullable**: No The high-level qualifier for the source table or view for this member.
SOURCE_TABLE	**Data type**: VARCHAR(128); VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: No The name of the source table or view for this member.
SOURCE_VIEW_QUAL	**Data type**: SMALLINT; **Nullable**: No Supports the view of physical tables by matching the similar column in the 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.
TARGET_OWNER	**Data type**: VARCHAR(128); **Nullable**: No The high-level qualifier for the target table or view for this member.
TARGET_TABLE	**Data type**: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: No The name of the target table or view for this member.
TARGET_CONDENSED	**Data type**: CHAR(1); **Nullable**: No A flag that indicates: **Y** For any given primary key value, the target table shows only one row. **N** All changes must remain to retain a complete update history. **A** The target table is a base aggregate or change aggregate tables.
TARGET_COMPLETE	**Data type**: CHAR(1); **Nullable**: No A flag that indicates: **Y** The target table contains a row for every primary key value of interest. **N** The target table contains some subset of rows of primary key values.

Table 94. Columns in the IBMSNAP_SUBS_MEMBR table (continued)

Column name	Description
TARGET_STRUCTURE	**Data type**: SMALLINT; **Nullable**: No The structure of the target table: **1** User table **3** CCD table **4** Point-in-time table **5** Base aggregate table **6** Change aggregate table **7** Replica **8** User copy **9** CCD table without a join of the IBMSNAP_UOW and CD tables
PREDICATES	**Data type**: VARCHAR(1024); **Nullable**: Yes Lists the predicates to be placed in a WHERE clause for the table in the TARGET_TABLE column. This WHERE clause creates a row subset of the source table. Predicates are recognized only when WHOS_ON_FIRST is set to S. The predicate cannot contain an ORDER BY clause because the Apply program cannot generate an ORDER BY clause. Aggregate tables require a dummy predicate followed by a GROUP BY clause. Because the Apply program uses these predicates for both full-refresh and change-capture replication, this column cannot contain predicates that involve columns in the CD or UOW table. Predicates that contain CD or UOW table references are stored in the UOW_CD_PREDICATES column.
MEMBER_STATE	**Data type**: CHAR(1); **Nullable**: Yes A flag that indicates what state the member is in: **N** (New) The member is new to this subscription set. Also, any members that were recently enabled will appear in this state. **L** (Loaded) The members of this subscription set have been loaded, but there has not yet been a change capture cycle. **S** (Synchronized) The member has been advanced from the new (N) state to the loaded (L) state, and is now synchronized with all the other subscription-set members that are in the synchronized state. When all members of a subscription set are in the synchronized state, change replication can occur at the subscription set level. **D** (Disabled) The member is disabled for this subscription set.

Table 94. Columns in the IBMSNAP_SUBS_MEMBR table (continued)

Column name	Description
TARGET_KEY_CHG	**Data type**: CHAR(1); **Nullable**: No A flag that indicates how the Apply program handles updates when, at the source table, you change the source columns for the target key columns of a target table: **Y** The Apply program updates the target table based on the before images of the target key column, meaning that the Apply program changes the predicate to the old values instead of the new. Make sure you have registered each before-image column of the target key so it is present in the CD table. For the corresponding registration entry in the register table, make sure the value in the CHG_UPD_TO_DEL_INS column is set to N. **N** The Apply program uses logic while processing updates and deletes that assume that the columns that make up the target key are never updated.
UOW_CD_PREDICATES	**Data type**: VARCHAR(1024); **Nullable**: Yes Contains predicates that include columns from the CD or UOW table that the Apply program needs only for change-capture replication, and not for full refreshes. During change-capture replication, the Apply program processes the predicates in this column and those in the PREDICATES column. During a full refresh, the Apply program processes only the predicates in the PREDICATES column.
JOIN_UOW_CD	**Data type**: CHAR(1); **Nullable**: Yes A flag that indicates whether the Apply program does a join of the CD and UOW tables when processing a user copy target table. This flag is needed when you define a subscription-set member with predicates that use columns from the UOW table that are not in the CD table. If the target table type is anything except user copy, then the Apply program uses a join of the CD and UOW tables when processing the member, and it ignores this column when processing the member. **Y** The Apply program uses a join of the CD and UOW tables when processing the member. **N** The Apply program does not use a join of the CD and UOW tables when processing the member; it reads changes only from the CD table. **NULL** The Apply program ignores this column when processing the member. If the target table is a user copy and the value in this column is null, then the Apply program does not do a join of the CD and UOW tables when processing the member.

Table 94. Columns in the IBMSNAP_SUBS_MEMBR table (continued)

Column name	Description
LOADX_TYPE	**Data type**: SMALLINT; **Nullable**: Yes The type of load for this member. The value in this column is used to override the defaults. This value is used only when the Apply invocation parameter **`loadxit`**=y is specified. **NULL** z/OS The LOAD from CURSOR function is used for this member. Linux UNIX Windows The ASNLOAD exit determines the most appropriate utility for this member (option 3, 4, or 5). **1** ASNLOAD is not used for this member. This effectively turns ASNLOAD option off for a particular subscription-set member even if you specified LOADX on startup. **2** A user-defined or user-modified ASNLOAD exit code is used. **3** The LOAD from CURSOR function is used for this member. Linux UNIX Windows **4** EXPORT and LOAD is used for this member. Linux UNIX Windows **5** EXPORT and IMPORT is used for this member. **6** The target table will not be loaded for this member. **Restriction:** Linux UNIX Windows The LOAD utility is not supported for range-clustered tables. To do a full refresh of a range-clustered table, you can either use the DB2 IMPORT utility or the Apply program to do a full refresh of the table through SQL.
LOADX_SRC_N_OWNER	**Data type**: VARCHAR(128); **Nullable**: Yes The user-created nickname owner. This value is required when all of the following conditions exist: • The LOAD from CURSOR function is used for this member (LOADX_TYPE is 3) • The target server is Linux, UNIX, or Windows • The source is not a nickname
LOADX_SRC_N_TABLE	**Data type**: VARCHAR(128); **Nullable**: Yes The user-created nickname table. This value is required when all of the following conditions exist: • The LOAD from CURSOR function is used for this member (LOADX_TYPE is 3) • The target server is Linux, 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 by using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

Table 95 provides a brief description of the columns in the IBMSNAP_SUBS_SET table.

Table 95. 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 95. Columns in the IBMSNAP_SUBS_SET table (continued)

Column name	Description
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.
STATUS	**Data type**: SMALLINT; **Nullable**: No A value that represents the work status for the Apply program after a given cycle: **-1** The replication failed. The Apply program backed out the entire set of rows it had applied, and no data was committed. If the startup parameter SQLERRCONTINUE = Y, the SQLSTATE that is returned to the Apply program during the last cycle is not one of the acceptable errors you indicated in the input file for SQLERRCONTINUE (*apply qualifier*.SQS). **0** The Apply program processed the subscription set successfully. If the startup parameter SQLERRCONTINUE = Y, the Apply program did not encounter any SQL errors that you indicated for the SQLERRCONTINUE startup parameter (in *apply qualifier*.SQS) and did not reject any rows. **1** The Apply program is processing the subscription set. **2** The Apply program is processing the subscription set in multiple cycles. It successfully processed a single logical subscription that was divided according to the MAX_SYNCH_MINUTES control column. **16** The Apply program processed the subscription set successfully and returned a status of 0; however, it encountered some SQL errors that you indicated for the SQLERRCONTINUE startup parameter (in *apply_qualifier*.SQS) and rejected some of the rows. See the *apply qualifier*.ERR file for details about the rows that failed. **Example**: You set SQLERRCONTINUE = Y and indicate that the allowable SQL state is 23502 (SQL code -407). A 23502 error occurs, but no other errors occur. The Apply program finishes processing the subscription set, and it sets the status to 16. On the next execution, a 23502 error occurs, but then a 07006 (SQL code -301) occurs. Now the Apply program stops processing the subscription set, backs out the entire set of rows it had applied, and sets the status to -1 (because no data was committed). **18** The Apply program is processing the subscription set in multiple cycles and returned a status of 2, meaning that it successfully processed a single logical subscription that was divided according to the MAX_SYNCH_MINUTES control column. However, it encountered some SQL errors that you indicated for the SQLERRCONTINUE startup parameter (in *apply_qualifier*.SQS) and rejected some of the rows. See the *apply_qualifier*.ERR file for details about the rows that failed.
LASTRUN	**Data type**: TIMESTAMP; **Nullable**: No The estimated time that the last subscription set began. The Apply program sets the LASTRUN value each time a subscription set is processed. It is the approximate time at the Apply control server when the Apply program begins processing the subscription set.

Table 95. Columns in the IBMSNAP_SUBS_SET table (continued)

Column name	Description
REFRESH_TYPE	**Data type**: CHAR(1); **Nullable**: No The type of scheduling that is used to prompt the Apply program to process this subscription set: **R** The Apply program uses time-based scheduling. It uses the value in SLEEP_MINUTES to determine when to start processing the subscription set. **E** The Apply program uses event-based scheduling. It checks the time value in the IBMSNAP_SUBS_EVENT table to determine when to start processing the subscription set. Before any replication (change capture or full refresh) can begin, an event must occur. **B** The Apply program uses both time-based and event-based scheduling. Therefore, it processes the subscription set based on either the time or event criteria.
SLEEP_MINUTES	**Data type**: INT; **Nullable**: Yes Specifies the time (in minutes) of inactivity between subscription set processing. The processing time is used only when REFRESH_TYPE is R or B. If the value of SLEEP_MINUTES is NULL, the Apply program will process the set continuously. The Apply program will process the set as often as possible, but will also process all other active subscription sets with the same Apply qualifier.
EVENT_NAME	**Data type**: CHAR(18); **Nullable**: Yes A unique character string used to represent the name of an event. Use this identifier to update the subscription events table when you want to trigger replication for a subscription set. The event name is used only when REFRESH_TYPE is E or B.
LASTSUCCESS	**Data type**: TIMESTAMP; **Nullable**: Yes The Apply control server timestamp for the beginning of the last successful processing of a subscription set.
SYNCHPOINT	**Data type**: VARCHAR(16) FOR BIT DATA; **Nullable**: Yes The Apply program uses this column to record its progress, indicating that it has processed data up to this synchpoint value for the subscription set.
SYNCHTIME	**Data type**: TIMESTAMP; **Nullable**: Yes The Apply program uses this column to record its progress, indicating that it has processed data up to this timestamp for the subscription set.
CAPTURE_SCHEMA	**Data type**: VARCHAR(128); **Nullable**: No The schema name of the Capture control tables that process the source for this subscription set.
TGT_CAPTURE_SCHEMA	**Data type**: VARCHAR(128); **Nullable**: Yes If the target table is also the source for another subscription set (such as an external CCD table in a multi-tier configuration or a replica table in an update-anywhere configuration), then this column contains the Capture schema that is used when the table is acting as a source.
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.

Table 95. Columns in the IBMSNAP_SUBS_SET table (continued)

Column name	Description
FEDERATED_TGT_SRVR	**Data type**: VARCHAR(18); **Nullable**: Yes The name of the federated remote server that is the target for the subscription set, which applies only to non-DB2 relational targets.
JRN_LIB	**Data type**: CHAR(10); **Nullable**: Yes System i This column, which applies only to System i Capture servers, is the library name of the journal that the source table uses.
JRN_NAME	**Data type**: CHAR(10); **Nullable**: Yes System i This column, which applies only to System i Capture servers, is the name of the journal used by a source table. An asterisk followed by nine blanks in this column means that the source table is currently not in a journal, in which case it is not possible to capture data for this source table.
OPTION_FLAGS	**Data type**: CHAR(4); **Nullable**: No Reserved for future options of SQL Replication. Currently this column contains the default value of NNNN.
COMMIT_COUNT	**Data type**: SMALLINT; **Nullable**: Yes A flag that indicates the type of processing that the Apply program performs for a subscription set: **NULL** This is the default setting for a read-only subscription set. The Apply program will process fetched answer sets for the *n* subscription-set members one member at a time, until all data has been processed, and then will issue a single commit at the end of the data processing for the whole set. The advantage of using this COMMIT_COUNT setting is that the processing might complete faster. *Integer not NULL* The Apply program processes the subscription set in a transactional mode. After all answer sets are fetched, the contents of the spill files will be applied in the order of commit sequence, ordering each transaction by the IBMSNAP_INTENTSEQ value order. This type of processing allows all spill files to be open and processed at the same time. A commit will be issued following the number of transactions specified in this column. For example, 1 means commit after each transaction, 2 means commit after each two transactions, and so on. An integer of 0 means that a single commit will be issued after all fetched data is applied. The advantage of using transactional mode processing is that the processing allows for referential integrity constraints at the target, and interim commits can be issued. Transaction-mode processing only changes the Apply program's behavior for sets with user-copy, point-in-time, and CCD target tables. Sets containing replica tables are always processed in transaction mode.
MAX_SYNCH_MINUTES	**Data type**: SMALLINT; **Nullable**: Yes A time-threshold limit to regulate the amount of change data to fetch and apply during a subscription cycle. The Apply program breaks the subscription set processing into mini-cycles based on the IBMSNAP_LOGMARKER column in the UOW or CCD table at the Capture server and issues a COMMIT at the target server after each successful mini-cycle. The limit is automatically recalculated if the Apply program encounters a resource constraint that makes the set limit unfeasible. MAX_SYNCH_MINUTES values that are less than 1 will be treated the same as a MAX_SYNCH_MINUTES value equal to null.

Table 95. Columns in the IBMSNAP_SUBS_SET table (continued)

Column name	Description
AUX_STMTS	**Data type**: SMALLINT; **Nullable**: No 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.
ARCH_LEVEL	**Data type**: CHAR(4); **Nullable**: No The architectural level of the replication control tables. This column identifies the rules under which a row was created. This level is defined by IBM. **0801** Version 8 or later SQL Replication **0803** Version 8 SQL Replication with enhanced support for Oracle sources **0805** Version 8 SQL Replication with support for DB2 for z/OS new-function mode

ASN.IBMSNAP_SUBS_STMTS 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 by 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 96 provides a brief description of the columns in the IBMSNAP_SUBS_STMTS table.

Table 96. Columns in the IBMSNAP_SUBS_STMTS table

Column name	Description
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No Uniquely identifies which Apply program processes the SQL statement or stored procedure.
SET_NAME	**Data type**: CHAR(18); **Nullable**: No The name of the subscription set that the SQL statement or stored procedure is associated with.

Table 96. Columns in the IBMSNAP_SUBS_STMTS table (continued)

Column name	Description
WHOS_ON_FIRST	**Data type**: CHAR(1); **Nullable**: No The following values are used to control the order of processing in update-anywhere replication scenarios. **F** (first) The target table is the user table or parent replica. The source table is the dependent replica and, in the case of update conflicts between the source table and the target table, the source table will have its conflicting transactions rejected. F is not used for read-only subscriptions. **S** (second) The source table is the user table, parent replica, or other source. The target table is the dependent replica or other copy and, in the case of update conflicts between the source table and the target table, the target table will have its conflicting transactions rejected. S is used for all read-only subscriptions.
BEFORE_OR_AFTER	**Data type**: CHAR(1); **Nullable**: No A value that indicates when and where the statement is issued: **A** The statement is executed at the target server after all of the answer-set rows are applied. **B** The statement is executed at the target server before any of the answer-set rows are applied. **S** The statement is executed at the Capture control server before opening the answer-set cursors. **G** Reserved for use by SQL Replication. **X** Reserved for use by SQL Replication.
STMT_NUMBER	**Data type**: SMALLINT; **Nullable**: No Defines the relative order of execution within the scope of the BEFORE_OR_AFTER column value.
EI_OR_CALL	**Data type**: CHAR(1); **Nullable**: No A value that indicates: **E** The SQL statement should be run as an EXEC SQL EXECUTE IMMEDIATE. **C** The SQL statement contains a stored procedure name to run as an EXEC SQL CALL.
SQL_STMT	**Data type**: VARCHAR(1024); **Nullable**: Yes One of the following values: **Statement** The SQL statement should run as an EXEC SQL EXECUTE IMMEDIATE statement if EI_OR_CALL is E. **Procedure** The eight-byte name of an SQL stored procedure, without parameters, or the CALL keyword that runs as an EXEC SQL CALL statement if EI_OR_CALL is C.

Table 96. 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.

Tables 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 97 describes the tables at the target server.

Table 97. Quick reference for target tables

Table name	Description
"Base aggregate table"	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.
"CCD targets" on page 77	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 455	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 456	A type of target table used for update-anywhere replication.
"User copy table" on page 456	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 98 provides a brief description of the columns in the base aggregate table.

Table 98. Columns in the base aggregate table

Column name	Description
user columns	The aggregate data that was computed from the source table.
IBMSNAP_LLOGMARKER	The current timestamp at the source server when the aggregation of the data in the source table began.

Table 98. Columns in the base aggregate table (continued)

Column name	Description
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 99 provides a brief description of the columns in the change aggregate table.

Table 99. Columns in the change aggregate table

Column name	Description
user key columns	The columns that make up the target key.
user nonkey columns	The nonkey data columns from the source table. The column names in this target table do not need to match the column names in the source table, but the data types must match.
user computed columns	User-defined columns that are derived from SQL expressions. You can use computed columns with SQL functions to convert source data types to different target data types.
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.

CCD targets

Consistent-change-data (CCD) target tables provide committed transactional data that can be read and used by other applications, for example InfoSphere DataStage. You might also use a CCD table to audit the source data or keep a history of how the data is used.

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

Important for Version 10.1 on Linux, UNIX, and Windows: If the source database is DB2 10.1 for Linux, UNIX, and Windows with multiple DB2 pureScale members, CCD targets are supported only if both the Capture and Apply programs are at Version 10.1 and the Capture **`compatibility`** parameter is set to 1001. With only a single DB2 pureScale member and **`compatibility`** set to 0801, CCD targets are supported even if the Apply program is at a version lower than 10.1.

By definition, a CCD table always includes the following columns in addition to the replicated columns from the source table:

Column	Description
IBMSNAP_INTENTSEQ	**Data type:** VARCHAR(16) FOR BIT DATA; **Nullable:** No A sequence number that uniquely identifies a change. This value is ascending in a transaction. z/OS The log sequence number (LRSN or RBA) of each update, delete, and insert.
IBMSNAP_OPERATION	**Data type:** CHAR(1); **Nullable:** No A flag that indicates the type of operation: I (INSERT), U (UPDATE), or D (DELETE).
IBMSNAP_COMMITSEQ	**Data type:** VARCHAR(16) FOR BIT DATA; **Nullable:** No A sequence number for each row within a transaction. z/OS The log sequence number (LRSN or RBA) of the source commit record.
IBMSNAP_LOGMARKER	**Data type:** TIMESTAMP; **Nullable:** No The approximate time that the data was committed.

When you create a noncomplete (COMPLETE=N) CCD table with the ASNCLP command-line program or Replication Center, you can specify additional auditing columns. The following table describes these columns:

Column	Description
IBMSNAP_AUTHID	**Data type:** VARCHAR(30); **Nullable:** Yes The user ID that updated the source table. z/OS This column is the primary authorization ID.
IBMSNAP_AUTHTKN	**Data type:** VARCHAR(30); **Nullable:** Yes The authorization token that is associated with the transaction. z/OS The correlation ID (normally a job name) that ran the source update.
IBMSNAP_PLANID	**Data type:** VARCHAR(8); **Nullable:** Yes z/OS The plan name that is associated with the transaction. This column will be null for DB2 for Linux, UNIX, and Windows.
IBMSNAP_UOWID	**Data type:** VARCHAR(16) FOR BIT DATA; **Nullable:** Yes The unit-of-work (UOW) identifier from the log record for a row. z/OS The unit-of-work identifier, sometimes called the unit-of-recovery ID (URID) of the transaction.

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 100 provides a brief description of the columns in the point-in-time table.

Table 100. Columns in the point-in-time table

Column name	Description
user key columns	The columns that make up the target key.

Table 100. Columns in the point-in-time table (continued)

Column name	Description
user nonkey columns	The nonkey data columns from the source table or view. The column names in this target table do not need to match the column names in the source table, but the data types must match.
user computed columns	User-defined columns that are derived from SQL expressions. You can use computed columns with SQL functions to convert source data types to different target data types.
IBMSNAP_LOGMARKER	The approximate commit time at the Capture control server. This column is null following a full refresh.

Replica table

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 101 provides a brief description of the columns in the replica table.

Table 101. Columns in the replica table

Column name	Description
user key columns	The columns that make up the target key, which must be the same primary key as the master table.
user nonkey columns	The nonkey data columns from the source table. The column names in this target table do not need to match the column names in the source table, but the data types must match.

User copy table

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 102 provides a brief description of the columns in the user copy table.

Table 102. Columns in the user copy table

Column name	**Description**
user key columns	The columns that make up the target key.
user nonkey columns	The nonkey data columns from the source table or view. The column names in this target table do not need to match the column names in the source table, but the data types must match.
user computed columns	User-defined columns that are derived from SQL expressions. You can use computed columns with SQL functions to convert source data types to different target data types.

Appendix A. UNICODE and ASCII encoding schemes for SQL replication (z/OS)

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 by 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_*xxxx*) on the same control server must use the same encoding scheme.
- Other control tables can use any encoding scheme.

Setting encoding schemes

To specify the proper encoding scheme for tables, modify the SQL that is used to generate the tables.

It is recommended that you stop the Capture and Apply programs before you change the encoding scheme of existing tables.

Note: The *DB2 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 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 by using the ASNCLP or Replication Center, specify the proper encoding scheme.

Appendix B. 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:

Windows
: sqllib\samples\repl

Linux 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.

Appendix C. How the Capture program processes journal entry types for SQL replication (System i)

The following table describes how the Capture program processes different journal entry types.

Table 103. Capture program processing by journal entry

Journal code[1]	Entry type	Description	Processing
C	CM	Set of record changes committed	Insert a record in the UOW stable.
C	RB	Rollback	No UOW row inserted.
F	AY	Journaled changes applied to physical file member	Issue an ASN2004 message and full refresh of file.
F	CE	Change end of data for physical file	Issue an ASN2004 message and full refresh of file.
F	CR	Physical file member cleared	Issue an ASN2004 message and full refresh of file.
F	EJ	Journaling for physical file member ended	Issue an ASN200A message and full refresh of the file. A full-refresh occurs whenever the Capture program reads an EJ journal entry, regardless of whether the user or the system caused journaling to end. Refer to the appropriate System i documentation for information about implicit end-journal events for a file.
F	IZ	Physical file member initialized	Issue an ASN2004 message and full refresh of file.
F	MD	Member removed from physical file (DLTLIB, DLTF, or RMVM)	Issue an ASN200A message and attempt a full refresh.
F	MF	Storage for physical file member freed	Issue an ASN200A message and full refresh of file.
F	MM	Physical file containing member moved (Rename Object (RNMOBJ) of library, Move Object (MOVOBJ) of file)	Issue an ASN200A message and attempt a full refresh.
F	MN	Physical file containing member renamed (RNMOBJ of file, Rename Member (RNMM))	Issue an ASN200A message and attempt a full refresh.
F	MR	Physical file member restored	Issue an ASN2004 message and full refresh of file.
F	RC	Journaled changes removed from physical file member	Issue an ASN2004 message and full refresh of file.
F	RG	Physical file member reorganized	If the RRN of the source table is being used as the replication key, issue an ASN2004 message and full refresh of file.

Table 103. 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:

1. The following values are used for the journal codes:

C	Commitment control operation
F	Database file operation
J	Journal or journal receiver operation
R	Operation on specific record

2. The R-UP image and the R-UB image form a single UPD record in the CD table if the PARTITION_KEYS_CHG column in the register table is N. Otherwise, the R-UB image inserts a DLT record in the CD table and the R-UP image inserts an ADD record in the CD table.
3. The R-UR image and the R-BR image form a single UPD record in the CD table if the PARTITION_KEYS_CHG column in the register table is N. Otherwise, the R-BR image inserts a DLT record in the CD table and the R-UR image inserts an ADD record in the CD table.

All other journal entry types are ignored by the Capture program.

Contacting IBM

You can contact IBM for customer support, software services, product information, and general information. You also can provide feedback to IBM about products and documentation.

The following table lists resources for customer support, software services, training, and product and solutions information.

Table 104. IBM resources

Resource	Description and location
IBM Support Portal	You can customize support information by choosing the products and the topics that interest you at www.ibm.com/support/entry/portal/Software/Information_Management/InfoSphere_Information_Server
Software services	You can find information about software, IT, and business consulting services, on the solutions site at www.ibm.com/businesssolutions/
My IBM	You can manage links to IBM Web sites and information that meet your specific technical support needs by creating an account on the My IBM site at www.ibm.com/account/
Training and certification	You can learn about technical training and education services designed for individuals, companies, and public organizations to acquire, maintain, and optimize their IT skills at http://www.ibm.com/software/sw-training/
IBM representatives	You can contact an IBM representative to learn about solutions at www.ibm.com/connect/ibm/us/en/

Federation, replication, and event publishing products support

For support, go to:

- IBM InfoSphere Federation Server
  www.ibm.com/software/data/integration/support/federation_server/
- IBM InfoSphere Replication Server
  www.ibm.com/software/data/integration/support/replication_server/
- IBM InfoSphere Data Event Publisher
  www.ibm.com/software/data/integration/support/data_event_publisher/

Classic products support

For support, go to:

- IBM InfoSphere Classic Federation Server for z/OS
  www.ibm.com/software/data/integration/support/classic_federation_server_z/

- IBM InfoSphere Classic Replication Server for z/OS
  www.ibm.com/software/data/infosphere/support/replication-server-z/
- IBM InfoSphere Classic Data Event Publisher for z/OS
  www.ibm.com/software/data/integration/support/data_event_publisher_z/
- IBM InfoSphere Data Integration Classic Connector for z/OS
  www.ibm.com/software/data/integration/support/data_integration_classic_connector_z/

Providing feedback

The following table describes how to provide feedback to IBM about products and product documentation.

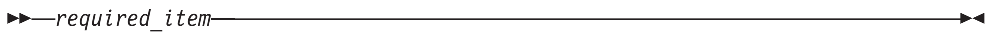
Table 105. Providing feedback to IBM

Type of feedback	**Action**
Product feedback	You can provide general product feedback through the Consumability Survey at www.ibm.com/software/data/info/consumability-survey
Documentation feedback	To comment on the information center, click the Feedback link on the top right side of any topic in the information center. You can also send comments about PDF file books, the information center, or any other documentation in the following ways: • Online reader comment form: www.ibm.com/software/data/rcf/ • E-mail: comments@us.ibm.com

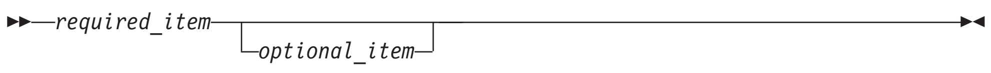
How to read syntax diagrams

The following rules apply to the syntax diagrams that are used in this information:

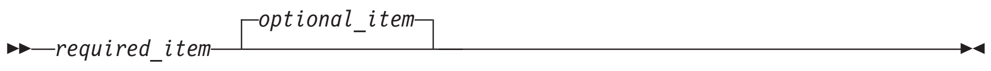
- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol indicates that a syntax diagram is continued from the previous line.
  - The --->< symbol indicates the end of a syntax diagram.
- Required items appear on the horizontal line (the main path).



- Optional items appear below the main path.

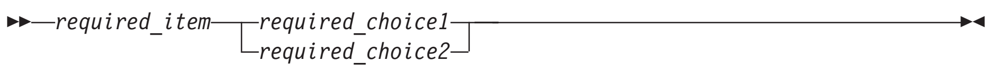


  If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

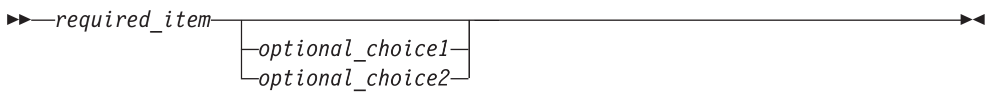


- If you can choose from two or more items, they appear vertically, in a stack.

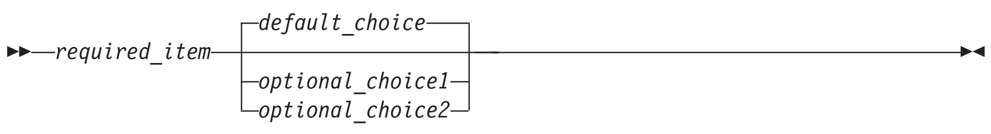
  If you must choose one of the items, one item of the stack appears on the main path.



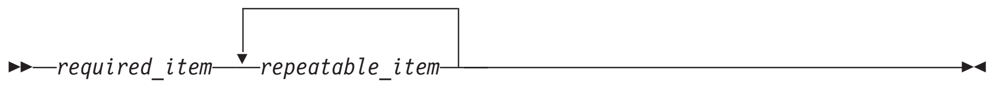
  If choosing one of the items is optional, the entire stack appears below the main path.



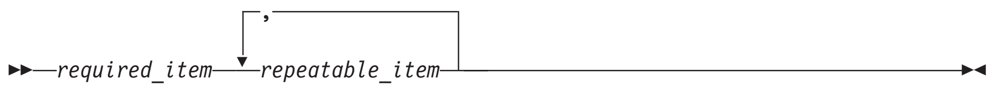
  If one of the items is the default, it appears above the main path, and the remaining choices are shown below.



- An arrow returning to the left, above the main line, indicates an item that can be repeated.

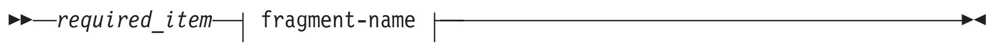


If the repeat arrow contains a comma, you must separate repeated items with a comma.

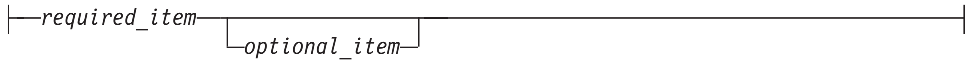


A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Sometimes a diagram must be split into fragments. The syntax fragment is shown separately from the main syntax diagram, but the contents of the fragment should be read as if they are on the main path of the diagram.



Fragment-name:



- Keywords, and their minimum abbreviations if applicable, appear in uppercase. They must be spelled exactly as shown.
- Variables appear in all lowercase italic letters (for example, **`column-name`**). They represent user-supplied names or values.
- Separate keywords and parameters by at least one space if no intervening punctuation is shown in the diagram.
- Enter punctuation marks, parentheses, arithmetic operators, and other symbols, exactly as shown in the diagram.
- Footnotes are shown by a number in parentheses, for example (1).

Notices and trademarks

This information was developed for products and services offered in the U.S.A.

Notices

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 character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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

Trademarks

Trademarks

Index

Printed in USA

SC19-3638-00

Spine information:

IBM InfoSphere Data Replication **Version 10.1.3** **SQL Replication Guide and Reference** IBM