Administration: Backup, Restore, and Data Recovery

SAP Sybase IQ 16.0 SP03

DOCUMENT ID: DC01759-01-1603-01
LAST REVISED: December 2013

Contents

Data Backup and Recovery

Regular and frequent backups are your only protection against database or device failure. Consider the performance implications of various backup options, and create an appropriate backup plan.

Backup plans depend on system load factors, database size, number of updates, and the relative importance of backup and recovery times. The length of time your organization can function without access to the data in your database determines the maximum recovery time. Run a full backup on a new database to provide a base point, then perform full and incremental backups on a fixed schedule. Consider time and storage requirements. Balance the time it takes to create the backup with the time it takes to restore the data.

Database Backup

The BACKUP DATABASE command backs up an SAP® Sybase® IQ database to one or more archive devices. Basic syntax identifies the type of backup you want to perform, and a **TO** clause that directs the output to an archive device:

```
BACKUP DATABASE
   [ backup-option … ]
TO archive_device
```

You must be connected to the database to back it up. You cannot use the BACKUP DATABASE command to specify another database.

In a multiplex environment, execute all BACKUP DATABASE and RESTORE DATABASE commands on the coordinator.

Backup Data

BACKUP DATABASE issues a CHECKPOINT before starting the backup, then backs up the catalog store. Data that is not committed when this initial checkpoint occurs is not included in the backup. A second checkpoint occurs at the end of backup. Data that is committed while the backup is in progress is included in subsequent backups.

Backups run concurrently with most other read-write operations, except those that affect the structure of the database. You cannot issue a CHECKPOINT or change the database metadata while a backup is in progress. If a system or media failure occurs during backup, you cannot restore uncommitted transactions.

Make a copy of the `params.cfg` file and save the contents of the SYSDBFILE and SYSDBSPACE system views. SAP Sybase IQ does not back up the temporary store (*db-name*`.iqtmp`) and `params.cfg`, but does back up the metadata and other information that is necessary for re-creating the temporary store structure.

Data Distribution

BACKUP DATABASE always makes a full backup of the catalog store on the first archive device, and then backs up the data from the IQ store in parallel across all of the devices you specify. Blocks are not evenly distributed across archive media. Depending on the processing speed of individual threads, you may have more blocks on one device than others.

SAP Sybase IQ backs up only those recoverable database blocks that are actually in use at the time of the backup. Free blocks are not backed up. Sets of files must be restored in the order in which they were backed up.

Backup Options

Dbspaces and dbfiles can be read-only (RO), read-write (RW), online, or offline. You can restrict **FULL**, **INCREMENTAL-SINCE-FULL**, or **INCREMENTAL** backups to the read-write files in the IQ main store (*db-name*`.iq`). The backed up files are selected when the backup command checks the read-write status in the catalog.

A backup can back up a set of read-only dbspaces and read-only files. The read-only dbspaces or files must belong to the IQ main store. The backed up files are user selected.

If you use symbolic links for raw device names, as recommended, make sure the system backup utility follows the symbolic link and backs up the device.

Device Limits

Keep backup commands small. Large numbers of devices increase I/O and hardware contention. To saturate CPU usage, use roughly 1 device per core; on faster systems, use up to 2 devices per core. Use 36 or fewer **TO** clauses.

Failure and Recovery

- If a backup fails during the initial or final CHECKPOINT, normal CHECKPOINT recovery occurs.
- If a backup fails between the initial or final CHECKPOINT, the backup is rolled back. If the system fails between the initial and final CHECKPOINTS, use an older backup to restore the database.
- If the system fails during the final CHECKPOINT after a full backup, restore the database from the backup you just created.

Note: See *Reference: Statements and Options > SQL Statements > BACKUP DATABASE* for parameter descriptions, usage, and required permissions.

See also

- *Database Validation* on page 3
- *Performance Options* on page 4
- *Archive Devices* on page 6
- *Queries, Utilities, and Procedures* on page 13

- *Backup Scenarios* on page 15
- *Previous Backups* on page 22
- *Backup Log* on page 24

Database Validation

Although BACKUP DATABASE ensures that the database is in a usable state, RESTORE DATABASE does not check for inconsistencies in restored data. Validate the database before a backup to ensure that the database you restore is stable.

The **sp_iqcheckdb** stored procedure is the interface to the database consistency checker (DBCC), which performs database verification. DBCC has different verification modes that perform increasing degrees of consistency checking. Run **sp_iqcheckdb** before a backup, or whenever you suspect a problem with the database. On multiplex, run **sp_iqcheckdb**only on a write server.

Set the database **DBCC_LOG_PROGRESS** option to write progress messages to the message file as **sp_iqcheckdb** executes.

Table 1. DBCC Verification Modes

Verification Mode	Description
Check Database Mode	Performs an internal consistency check on all IQ indexes and checks that each database block has been allocated correctly. Syntax: `sp_iqcheckdb 'check database'`
Verify Database Mode	Reads all data pages and can detect all types of allocation problems and all types of index inconsistencies. Syntax: `sp_iqcheckdb 'verify database'`
Database Allocation Mode	Checks that each database block is allocated correctly according to the internal physical page mapping structures. Syntax: `sp_iqcheckdb 'allocation database'`

See also

- *Database Backup* on page 1
- *Performance Options* on page 4
- *Archive Devices* on page 6
- *Queries, Utilities, and Procedures* on page 13

- *Backup Scenarios* on page 15
- *Previous Backups* on page 22
- *Backup Log* on page 24

Performance Options

BLOCK FACTOR specifies the number of blocks to write to the archive device at one time. This parameter also controls the amount of memory used for buffers during the backup, and has a direct impact on backup performance. The effects of the block factor are a function of disk subsystem speed, tape speed, and processor speed.

Performance also depends on your operating system and on the block size specified when the database was created. The default SAP Sybase IQ page size of 128KB for newly created databases results in a default block size of 8192 bytes.

Table 2. BLOCK FACTOR Settings by Platform

Platform	
UNIX-like operating systems	Set **BLOCK FACTOR** to at least 25 (default). With this setting, BACKUP can buffer data for most tape drives, with enough data in memory that drives are kept busy constantly throughout the back-up. On AIX, use the System Management Interface Tool (SMIT) to display and change the current block mode of any tape device that you plan to use for backups.
Windows	On Windows, the default **BLOCK FACTOR** is computed based on the database block size. The default usually achieves maximum throughput on Windows. Because of the way Windows handles tape devices, increasing the **BLOCK FACTOR** may not lead to faster backups.

See your platform operating system documentation for information about your platform's optimal I/O size and block factor.

Database Verification Concurrency Issues

During database verification, **sp_iqcheckdb** reads every database page in use, which consumes most of the database server's time. I/O remains as efficient as possible, but other concurrent activities may run more slowly than usual.

To limit DBCC CPU use, set the **resources resource-percent** parameter, which controls the number of threads with respect to the number of CPUs. If **resource-percent** = 100 (the default value), there is one thread per CPU. If **resource-percent** > 100, then there are more threads than CPUs, which might increase performance for some machine configurations.

Error Checking

Setting the **CRC** parameter to OFF deactivates 32-bit cyclical redundancy checking on a per-block basis, which improves the speed of backup and restore operations.

With **CRC** set to ON (default), the checksums computed on backup are verified during any subsequent RESTORE operation, affecting the performance of both commands. If you turn off this checking, remember that you sacrifice a greater assurance of accurate data in exchange for faster performance.

Comments

The **WITH COMMENT** parameter lets you specifies a string as long as 32KB as part of the header information for the backup archive. If you omit this option, BACKUP enters a NULL. You can view the comment string by executing a **RESTORE DATABASE... FROM... CATALOG ONLY** or by displaying the backup log, `backup.syb`.

Catalog Store Size

BACKUP makes a full backup of the catalog store at the start of every backup, both full and incremental. Ordinarily, the catalog store is quite small, containing only the system tables, metadata, and other information that SAP Sybase IQ needs to manage your database. However, you can create non-IQ tables in the catalog store.

To improve performance keep any non-SAP Sybase IQ data in a separate SAP Sybase SQL Anywhere®-only database, rather than in the catalog store. BACKUP copies only the latest committed version of the database. Other version pages used by open transactions are not backed up.

Spooling Backup Data

You may find that it is faster and more efficient to create backups on disk, and then spool them onto tape for archival storage. If you choose this approach, unspool the data onto disk before restoring it.

See also

- *Database Backup* on page 1
- *Database Validation* on page 3
- *Archive Devices* on page 6
- *Queries, Utilities, and Procedures* on page 13
- *Backup Scenarios* on page 15
- *Previous Backups* on page 22
- *Backup Log* on page 24

Archive Devices

Specific information about writing to different archive devices, including tape, disk, read-only hardware, and third-party products.

See also

- *Database Backup* on page 1
- *Database Validation* on page 3
- *Performance Options* on page 4
- *Queries, Utilities, and Procedures* on page 13
- *Backup Scenarios* on page 15
- *Previous Backups* on page 22
- *Backup Log* on page 24

Tape Backups

A tape set consists of one or more backup tapes produced on a given archive device. The first tape set you specify must be large enough to hold the full backup of the catalog store, including any non-SAP Sybase IQ data in the catalog store.

Digital Linear Tape (DLT) is the preferred magnetic tape data storage medium. If it is supported for your platform, use DLT for tape backups. BACKUP can also support 4mm and 8mm Digital Data Storage (DDS) tape drives, and stacker devices that support multiple tape devices.

BACKUP does not support jukeboxes or robotic loaders. Use a third-party media manager for these items.

Large Database Backups

Use multiple tape drives for large database backups. The **TO** clause identifies each tape drive device. The **SIZE** option sets the data capacity for each tape. The **STACKER** option indicates a backup to a multi-tape stacker device, and the number of tapes in the device. All tapes in a given stacker device must be the same size.

Tape Devices on UNIX-Like Operating Systems

BACKUP does not support fixed-length block-mode tape devices on any UNIX-like OS. To check the device block-size status, run this shell command:

```
mt -f <tape device> status
```

If the command returns a block-size greater than 0, the device is set for fixed-length blocks. To change the tape drive to variable-length blocks, use this or other appropriate OS shell command:

```
mt -f <tape device> defblksize 0
```

Table 3. Non-rewinding Tape Device Commands

Platform	Description
Solaris	Insert the letter *n* for no rewind after the device name: `/dev/rmt/0n`
AIX	Set the numeric value that follows the logical name of the tape drive for an no rewind setting: `/dev/rmt0.1` See the AIX documentation for specific settings.
HP-UX	Use **0m** to specify the default tape mechanism and *n* for no rewind: `/dev/rmt/0mn`

Note: See your platform documentation for specific instructions and options.

Tape Devices on Windows

Windows does not support *rewind* or *no rewind* devices. SAP Sybase IQ requires variable-length devices, but does perform additional processing to accommodate fixed-length tape I/O on Windows.

Because SAP Sybase IQ does not support tape partitioning, do not use another application to format tapes for BACKUP or RESTORE operations. On Windows, the first tape device is `\\.\tape0`, the second is `\\.\tape1`, and so on.

Note: SAP Sybase IQ treats the leading backslash in a string as an escape character, when the backslash precedes an n, an x, or another backslash. When you specify backup tape devices, double each backslash required by the Windows naming conventions.

To indicate the first tape device, use `\\\\.\\tape0`, `\\\\.\\tape1` for the second device, and so on. If you omit the extra backslashes, or otherwise misspell a tape device name, and write a name that is not a valid tape device on your system, SAP Sybase IQ interprets this name as a disk file name.

Tape Backup Size

Use the **SIZE** parameter to set the data capacity for each tape. The **SIZE** value can be less than or greater than the default tape size. If you omit the **SIZE** parameter for an unattended backup, the entire backup must fit on one tape.

Table 4. Default Tape Sizes

Platform	Default Tape Size
UNIX-like operating systems	None

Platform	Default Tape Size
Windows	1.5GB

Windows – the **SIZE** parameter must be a multiple of 64. Other values are rounded down to a multiple of 64.

The number of tapes supplied along with the **SIZE** clause determines whether there is enough space to store the backed-up data. Use separate **TO** clauses to set the **SIZE** value, in kilobytes (KB), for each tape:

```
TO '/dev/rmt/0n' SIZE 10000000
TO '/dev/rmt/2n' SIZE 15000000
```

Use separate lines for each **TO** clause, not a comma-delimited list.

During backup, when the amount of information exceeds the tape capacity or reaches the value specified by the **SIZE** parameter, BACKUP closes the current tape. For attended and unattended backups that specify the **SIZE** and **STACKER** parameters, SAP Sybase IQ waits for the stacker device to auto-load the next tape, then resumes the backup.

If the backup exceeds the tape capacity in an attended backup without a **STACKER** device, BACKUP prompts you to mount a new tape. If the backup exceeds the tape capacity in a unattended backup without a **STACKER** device, the backup fails.

Rewind Tapes

SAP Sybase IQ does not rewind tapes before using them. Set the tapes to the correct starting point before you put them in the tape device. If you use a rewinding device, tapes are rewound after backup. If your tape device automatically rewinds tapes, position the tape so that a subsequent backup does not overwrite any information on the tape.

Wait for Tape Devices

If SAP Sybase IQ cannot open an archive device, the server waits for 10 seconds and tries again. These attempts continue indefinitely, until the operation succeeds or is terminated. A message is written to the server `.stderr` file. There is no console notification that the server cannot open the archive device.

Restoring Tape Backups

Position the tape to the start of the IQ data. RESTORE does not reposition the tape for you. Use the same number of tape drives for the restore you used to create the backup.

See also

- *Disk Backups* on page 9
- *Read-Only Hardware* on page 10
- *Third-Party Products* on page 12

Disk Backups

All disk backups write to the file system; raw disk backup is not supported. All disks on a redundant array of independent devices (RAID) device are treated as a single device.

BACKUP appends a suffix to the *archive_device* name to assign file names to disk backup files. The suffix consists of "." followed by a number that increases by one for each new file. For example, if you specify `/iqback/mondayinc` as the *archive_device*, the backup files are `/iqback/mondayinc.1`, `/iqback/mondayinc.2`, and so on. This convention allows you to store as large a backup as you need, while allowing you control over the file size; see the **SIZE** option for details. To accommodate this convention, your file system must support long file names.

Disk File Location

BACKUP does not create missing directories. If you try to start a backup in a directory that does not exist, the backup fails.

Do not use relative path names for disk file locations. BACKUP interprets the path name as relative to the location where the server was started, which you may not be able to identify with certainty when you do a backup. If there is data in other directories along the path, you may not have enough disk space to perform the backup.

Disk Backup Size

BACKUP determines how much disk space is required to back up your database. If there is not enough disk space available, the backup fails before writing any data.

Use the optional **SIZE** parameter to define backup file sizes. The **SIZE** value can be less than or greater than the default disk size. Backups that do not include **SIZE** parameters proceed until the backup is complete (backup is less than disk capacity) or the disk is full.

Platform	Default Disk Size
UNIX-like operating systems	2GB
Windows	1.5GB

Windows – the **SIZE** must be a multiple of 64. Other values are rounded down to a multiple of 64.

The SIZE value is expressed kilobytes (KB). **SIZE** does not limit the number of bytes per device. **SIZE** limits the file size.

During backup, when the amount of information written to a given file exceeds the default disk size, or reaches the value specified by the **SIZE** parameter, BACKUP closes the current file and creates another file of the same name, with the next ascending number appended to the file name. For example, `bkup1.dat1.1`, `bkup1.dat1.2`, `bkup1.dat1.3`.

An unattended backup that runs out disk space fails. An attended backup closes all open backup files and prompts you to free up additional disk space (8KB minimum). When the additional space becomes available, the backup resumes with new backup files.

Previous Backups

BACKUP overwrites existing disk files of the same name. To retain a previous backup, use different file or path names for the archive devices, or move the old backup to another location.

Restoring Disk Backups

If you back up to disk, then move the backups to tape, move the backups back to disk with the same file names you used when you created the tape backup. SAP Sybase IQ cannot restore disk backups from the tapes where you moved the tape backups. When you restore from disk, you must specify the same number of archive devices (disk files) for the restore as were used to create the backup.

See also

- *Tape Backups* on page 6
- *Read-Only Hardware* on page 10
- *Third-Party Products* on page 12

Read-Only Hardware

Write-once read-many (WORM) disk arrays provide read-only hardware functionality that allows normal read-write use of the disks until data is frozen. Disks can be frozen for an indefinite or fixed-retention period at the volume or file level. Frozen data cannot be modified, and the retention period can be extended, but never decreased.

Read-only hardware functionality is not limited to WORM disk array hardware; you can also remove write privilege from a raw device or file system file after the dbspace is altered to read-only. To create and update an archive on read-only hardware:

1. *Creating an Archive*

   Assume that the database consists of a single catalog store dbspace named db.db with three main dbspaces: A, B and C.
2. *Create New Dbspaces*

   After creating the archive, create new dbspaces .
3. *Examine Archived Data*

   Examine the archived database as of time t0.
4. *Update the Working Archive*

   If a long time (for example, months or even years) have passed since time t0, you may upgrade the db.db0.working as long as ALTER DATABASE UPGRADE modifies no objects in the IQ main store.
5. *Create More Archives*

Create a new archive at time t1.

See also

- *Tape Backups* on page 6
- *Disk Backups* on page 9
- *Third-Party Products* on page 12

Creating an Archive

Assume that the database consists of a single catalog store dbspace named `db.db` with three main dbspaces: `A`, `B` and `C`.

1. At time `t0`, alter all three main dbspaces to read-only.
2. Copy `db.db` to `db.db0`, either by shutting the database down and copying `db.db` or using **dbbackup** to make a copy while the database is still running.
3. Freeze dbspaces `A`, `B` and `C` at the hardware level. Store `db.db0` in an immutable form, perhaps by storing it in a file system file on the WORM device and freezing it.

These steps archive the database as of time `t0` in an immutable form.

Create New Dbspaces

After creating the archive, create new dbspaces .

1. Create two new main dbspaces, `D` and `E`.
2. Continue using the database `db.db` as a production database.
   - The database objects (tables, indexes, and so on) that existed as of time `t0` may have changed so that `db.db` does not equal `db.db0`.
   - The database `db.db` continues to read data from dbspaces `A`, `B`, and `C` as long as the tables that existed at time `t0` exist and as long as they contain some unmodified rows of data that existed as of time `t0`.
   - Even if or when this is no longer true, `db.db` continues to open `A`, `B` and `C` unless they are dropped from `db.db`, which is allowed only if they are empty from `db.db`'s point of view.

Examine Archived Data

Examine the archived database as of time `t0`.

1. Copy the archived read-only `db.db0` to a read-write file `db.db0.working`.
2. Start `db.db0.working`.

   As long as the server name `db.db0.working` does not conflict with the production system `db.db`, there is no need to stop the production system. The `db.db0.working` server opens `A`, `B`, `C`, and `D` in read-only mode; the catalog opens in read-write mode.

On a UNIX-like OS, this does not interfere with db.db's use of these files. Windows returns a sharing violation.

3. Create user inv; an investigator who wants to examine the archived database.
4. Grant inv RESOURCE permission to create views, stored procedures, global or local temporary tables or any other structures necessary for the investigation.

   db.db0 as well as A,B, and C remain unchanged.

Update the Working Archive

If a long time (for example, months or even years) have passed since time t0, you may upgrade the db.db0.working as long as **ALTER DATABASE UPGRADE** modifies no objects in the IQ main store.

1. The temporary dbspaces that existed as of time t0 are not required to start db.db0.working. Use the server startup switch **-iqnotemp** to start db.db0.working.
2. Drop and create new temporary dbspaces or use the temporary space created by the **-iqnotemp** parameter.

Create More Archives

Create a new archive at time t1.

1. Alter dbspaces D and E to read-only.
2. Copy db.db to db.db1.
3. Freeze D and E.
4. Save db.db1 in an immutable form.
5. Create new main dbspaces, for example, F and G.
6. Continue to use the production system db.db.

To use the archived databases db.db0 or db.db1, or even both simultaneously, copy db.db0 and db.db1 to a working file and start a server. Create an archive and follow this procedure to create any number of archived versions of db.db.

Third-Party Products

SAP Sybase IQ supports backup and restore operations with third-party products that support SAP Sybase databases.

To perform a backup or restore with a third-party product, issue the BACKUP DATABASE or RESTORE DATABASE statement as if you were using SAP Sybase IQ to perform the operation, with the following exceptions:

- For each *archive_device*, instead of specifying the actual device name, specify a string in the following format:

```
dll_name::vendor_specific_information
```

- Do not specify the **STACKER** or **SIZE** parameters.

The *dll_name* corresponds to a dynamic link library loaded at run time. The *dll_name* can be from 1 to 30 bytes long, and can contain only alphanumeric and underscore characters. It must be the same for each *archive_device*.

vendor_specific_information varies by product, and can differ for each *archive_device*. The total string (including *dll_name*:: *vendor_specific_information*) can be up to 255 bytes long.

BACKUP DATABASE automatically passes vendor information to the third-party program. When you request a third-party backup, BACKUP DATABASE writes this information in the backup header file, and moves the header file on the first tape or disk file actually created for each *archive_device*.

Note: See the *Release Bulletin* for additional usage instructions or restrictions. Before using any third-party product to back up your database, make sure it is certified with SAP Sybase IQ. See the SAP Sybase Certification Reports for the SAP Sybase IQ product in *Technical Documents*.

See also

- *Tape Backups* on page 6
- *Disk Backups* on page 9
- *Read-Only Hardware* on page 10

Queries, Utilities, and Procedures

Queries, utilities, and procedures return information about the tablespaces, dbspaces, and dbfiles in your database.

The SYSDBFILE system view shows all the dbfiles in your database, including the catalog, message file, and dbfiles in the main and temporary dbspaces. To return dbfile and dbspace statistics, query the SYSDBFILE system view:

```
SELECT dbf.dbfile_name, f.*
FROM SYSFILE f, SYSDBFILE dbf
WHERE f.file_id=dbf.dbfile_id
```

Results are similar to this:

```
dbfile_name         file_id file_name                              dbspace_name   store_type lob_map
dbspace_id
------------------- ------- ------------------------------------ -------------- ---------- --------
----------
system                0 /dev/rdsk/SybaseIQ/demo/iqdemo.db   system               1 (NULL)           0
temporary            15 /temp/sqla0000.tmp                temporary            1 (NULL)          15
IQ_SYSTEM_MAIN      16384 iqdemo.iq                       IQ_SYSTEM_MAIN       2 (NULL)       16384
IQ_SYSTEM_TEMP      16385 iqdemo.iqtmp                    IQ_SYSTEM_TEMP       2 (NULL)       16385
IQ_SYSTEM_MSG       16386 iqdemo.iqmsg                    IQ_SYSTEM_MSG        2 (NULL)       16386
iq_main              16387 iqdemo_main.iq                     iq_main               2 (NULL)
16387
```

The `file_name` column in the SYSFILE system table for the SYSTEM dbspace is not updated during a restore. For the SYSTEM dbspace, the `file_name` column always reflects

the name when the database was created. The file name of the SYSTEM dbspace is the name of the database file.

See *Reference: Building Blocks, Tables, and Procedures > System Tables and Views > System View > Alphabetical List of System Views > SYSDBFILE System View*.

db_backupheader Utility

Reads the first backup archive, returns backup statistics and definitions.

Syntax:

```
db_backupheader [ path ] backup_file
```

db_backupheader is a command line utility. Output includes backup statistics, database definitions, and dbspace and dbfile specifics.

sp_iqdbspace Procedure

Returns details about each dbspace.

Syntax:

```
sp_iqdbspace [ dbspace-name ]
```

Results are similar to this:

```
DBSpaceName      DBSpaceType  Writable Online Usage TotalSize Reserve NumFiles NumRWFiles Stripingon StripeSize
BlkTypes           OkToDrop
---------------- ------------ -------- ------ ----- --------- ------- -------- ----------- ---------- ----------
---------------- --------
iq_main          MAIN         T        T      26    100M      200M          1           1 T         1K         1H,3254A
N
IQ_SYSTEM_MAIN   MAIN         T        T      22    100M      200M          1           1 T         1K         1H,2528F,32D,128M
N
IQ_SYSTEM_TEMP   TEMPORARY    T        T      3     25M       200M          1           1 T         1K         1H,64F,16A
N
```

See *Reference: Building Blocks, Tables, and Procedures > System Procedures > Alphabetical List of System Stored Procedures > sp_iqdbspace Procedure* for column definitions.

sp_iqfile Procedure

Returns details about the dbfiles in a dbspace.

Syntax:

```
sp_iqfile [ dbspace-name ]
```

Results are similar to this:

```
DBSpaceName     DBFileName      Path          SegmentType RWMode Online Usage DBFileSize Reserve StripeSize BlkTypes
FirstBlk  LastBlk  OkToDrop
--------------- --------------- ------------- ----------- ------ ------ ----- ---------- ------- ----------
----------------- --------- -------- ---------
IQ_SYSTEM_MAIN  IQ_SYSTEM_MAIN   iqdemo.iq      MAIN        RW     T      22    100M       200M    1K         1H,2528F,32D,
128M          1     12800  N
iq_main         iq_main         iqdemo_main.iq MAIN        RW     T      26    100M       200M    1K         1H,3254A
1045440   1058239  N
IQ_SYSTEM_TEMP  IQ_SYSTEM_TEMP   iqdemo.iqtmp   TEMPORARY   RW     T      3     25M        200M    1K         1H,64F,
16A               1     3200  N
```

See *Reference: Building Blocks, Tables, and Procedures > System Procedures > Alphabetical List of System Stored Procedures > sp_iqfile Procedure*.

Renaming Dbspaces

To move a database or dbspace, you need to know the name of every dbspace in the database when the backup was made. You can also run the following script in Interactive SQL. This script produces an output file that contains the set of **RENAME** clauses you can use, if you do not actually change the location of any files. You can substitute any new file locations, and use the resulting file in your RESTORE DATABASE statement.

```
-- Get dbspace and IQ file names and add
-- rename syntax including quotation marks

select 'rename' as 'restore ... rename' ,
dbf.dbfile_name as 'IQ file' , 'to' as 'to' ,
'''' + f.file_name + '''' as 'file_path'
from SYSFILE f, SYSDBFILE dbf
where f.store_type=2 and f.file_id=dbf.dbfile_id

 -- Send output to a file in proper format
 -- without delimiters or extra quotation marks

output to restore.tst delimited by '' quote '';

 -- This produces a restore.tst file like the following:
 -- rename IQ_SYSTEM_MAIN to '/dev/rdsk/c2t0d1s7'
 -- rename IQ_SYSTEM_TEMP to '/dev/rdsk/c2t1d1s7'
 -- rename IQ_SYSTEM_MSG to 'all_types.iqmsg'
```

Note: Because the database may not exist when you need to restore, you may want to run this script after you back up your database.

See also

- *Database Backup* on page 1
- *Database Validation* on page 3
- *Performance Options* on page 4
- *Archive Devices* on page 6
- *Backup Scenarios* on page 15
- *Previous Backups* on page 22
- *Backup Log* on page 24
- *Cache Dbspace Backup* on page 17

Backup Scenarios

Background information and code samples for routine, system-level, and virtual backups.

See also

- *Database Backup* on page 1
- *Database Validation* on page 3

Routine Backups

Code samples, background information, and tips for standard backup options.

FULL Backups

FULL backups make a complete copy of the database. The **FULL** keyword is optional. This example backs up a database to two tape devices:

```
BACKUP DATABASE
    TO '/dev/rmt/0'
    TO '/dev/rmt/1'
WITH COMMENT 'Jan 18 full backup of iquser'
```

This example restores the database:

```
RESTORE DATABASE 'iquser'
    FROM '/dev/rmt/0'
    FROM '/dev/rmt/1'
```

INCREMENTAL Backup

INCREMENTAL backups copy all transactions since the last backup of any type. To make an incremental backup, use the **INCREMENTAL** keyword. This example creates an incremental backup of the database onto one tape device:

```
BACKUP DATABASE
INCREMENTAL
    TO '/dev/rmt/0' SIZE 150
WITH COMMENT 'Jan 30 incremental backup of iquser'
```

The **SIZE** parameter specifies the maximum storage capacity per output device.

INCREMENTAL SINCE FULL Backups

INCREMENTAL SINCE FULL backups back up all changes to the database since the last full backup. This example backs up the database to two tape devices:

```
BACKUP DATABASE
INCREMENTAL SINCE FULL
TO '/dev/rmt/0' SIZE 10000000
TO '/dev/rmt/2' SIZE 15000000
```

All blocks changed since the last full backup are saved to separate tape devices.

Read-Only Files and Dbspaces

The **READWRITE** and **READONLY** keywords limit backups to read-write or read-only files or dbspaces. The read-write dbspaces and files must be SAP Sybase IQ dbspaces.

Back up a read-only dbspace to a tape device:

```
BACKUP DATABASE READONLY DBSPACES dsp1
TO '/dev/rmt/0'
```

Restore the dbspace:

```
RESTORE DATABASE READONLY DBSPACES dsp1
FROM '/dev/rmt/0'
```

Back up read-only files to `bkp.f1f2`:

```
BACKUP DATABASE READONLY FILES dsp1_f1, dsp1_f2
TO 'bkp.f1f2'
```

Back up read-only dbspaces and read-only files to `bkp.RO`:

```
BACKUP DATABASE READONLY DBSPACES dsp2, dsp3 READONLY FILES dsp4_f1,
dsp5_f2
TO 'bkp.RO'
```

See also

- *Cache Dbspace Backup* on page 17
- *System-Level Backups* on page 18
- *Virtual Backups* on page 20

Cache Dbspace Backup

If your installation uses the cache dbspace for direct-attached storage, follow best practices to avoid errors during backup.

- The cache dbspace must be online. An offline cache dbspace causes the backup to fail with error 1012034. For example:

  ```
  Msg 21, Level 14, State 0:
  SQL Anywhere Error -1012034: IQ store dbspace or dbfile IQDAS1 is
  unavailable. Backup will not be done.
  (dblib/db_backupID.cxx 859)
  ```

- The cache dbspace must be read-write. A read-only cache dbspace causes the backup to fail with error 1012058. For example:

  ```
  Msg 21, Level 14, State 0:
  SQL Anywhere Error -1012058: IQ cache dbspace IQDAS1 is read-only.
  Backup will not be done.
  (dblib/db_backupID.cxx 863)
  ```

- Backing up individual cache dbspace dbfiles results in an error. SAP Sybase IQ can reconstruct the cache dbspace, provided it sees a raw device, or a file of the correct length at the time the database starts. Before backing up, record the sizes of the files used to hold the cache dbspaces.
- Changing the path of a cache dbspace dbfile on a secondary node using the RENAME clause results in an error.

See also

- *Routine Backups* on page 16
- *System-Level Backups* on page 18
- *Virtual Backups* on page 20

System-Level Backups

The **BACKUP DATABASE** command is the most reliable method you can use to back up IQ data. If you prefer, you can perform a system-level backup, where you handle the entire data backup yourself. To ensure consistent catalog, metadata, and data, shut down the database before a system-level backup.

Follow the steps below when you perform a system-level backup. If you attempt to restore your database without these safeguards in place, you are likely to cause data loss or inconsistency, either from activity in the database while the system-level backup occurred, or from missing files.

1. *Shut Down the Database*

   Shut down your SAP Sybase IQ database before performing a system-level backup.
2. *Back Up the Correct Files*

   Back up required files and optional files.
3. *Restoring System-Level Backups*

   Shut down the database server. On a multiplex, shut down all the secondary servers as well as the coordinator.

See also

- *Routine Backups* on page 16
- *Cache Dbspace Backup* on page 17
- *Virtual Backups* on page 20

Shut Down the Database

Shut down your SAP Sybase IQ database before performing a system-level backup.

Ensure that no one starts the SAP Sybase IQ database until the system-level backup is complete.

Ensuring That the Database is Shut Down

The file protection status of the `.db` file is read-only when the database is shut down cleanly, and read-write when the database is in use. If you are writing a script to perform backups, make sure the script check the access mode of the file, to be sure that the database is shut down.

To ensure that a database remains shut down, the script can check the size of the `.iqmsg` file at the start and end of the script to make sure it has not changed. If the database was started while the script was running, the `.iqmsg` file is larger.

Back Up the Correct Files

Back up required files and optional files.

Required Files

- All `SYSTEM` dbspace files, typically named `dbname.db`, including any additional dbspaces in the catalog store, listed in `SYSDBSPACES`
- The transaction log file, which is required for system recovery, typically named `dbname.log`
- The `IQ_SYSTEM_MAIN` dbspace file and typically named `dbname.iq`
- Files for any additional dbspaces that have been added to the IQ main store

Save the lengths of the following files:

- The `IQ_SYSTEM_TEMP` dbspace file, typically named `dbname.iqtmp`
- Additional files that have been added to `IQ_SYSTEM_TEMP`

You are not required to back up temporary dbspaces. SAP Sybase IQ can reconstruct any temporary dbspace provided there is a file of the correct length at the time the database starts. Therefore, you may simply keep records of the sizes of the files or raw devices used to hold the temporary dbspaces.

Optional Files

Back up the ASCII message files such as `dbname.iqmsg`, the `$IQDIR16/logfiles/*.srvlog`, and `$IQDIR16/logfiles/*.stderr` files, even though these files are not required for a restore. If problems occur during a restore, the `.iqmsg` file contains information that proves that the database was shut down before the backup started.

These files may be useful in diagnosing the cause of the database failure you are recovering from. Make a copy before restoring, for use in later analysis.

If IQ message log wrapping is enabled, back up the `.iqmsg` file so that all messages are accessible if you need them for diagnostic purposes.

If message log archiving is enabled (the **IQMsgMaxSize** server option or the **-iqmsgsz** server startup switch is not equal to zero, and the **IQMsgNumFiles** server option or the **-iqmsgnum** server startup switch is not equal to zero), the server automatically backs up the message log archives. The maximum amount of message log that is archived is 128GB, which is sufficient in most cases.

Note: Before a server restart, you must back up the message log archives. After the server restarts, existing log archives are ignored and a new archive is created when the `dbname.iqmsg` file is full.

Keeping Your Backup List Updated

It is critical to add to your system backup specification any dbspaces that are added to the database, whether they are in `SYSTEM`, `IQ_SYSTEM_MAIN`, or `IQ_SYSTEM_TEMP`.

To ensure that you are backing up all the files you need, use a script for system-level backups. In the script, before starting the backup, compare a select from *SYSFILE* (for the system dbspaces) and from *SYSIQFILE* (for the IQ dbspaces) to a list of dbspaces known to be in the system backup specification.

Raw Devices and Symbolic Links

If your database files are on raw devices, be sure your system backup is backing up the raw device contents, not just the name of the device in `/dev/*`.

If you use symbolic links for raw device names, as recommended, be sure the system backup utility follows the symbolic link and backs up the device.

Restoring System-Level Backups

Shut down the database server. On a multiplex, shut down all the secondary servers as well as the coordinator.

Before you restore:

- Review the table of contents of the backup to ensure that all required SAP Sybase IQ files are present. The file list depends on your application.
- For temporary dbspace files, ensure that all files or raw devices are present with the correct file names (or symbolic links) and lengths. Contents of temporary dbspace files are irrelevant until the database restarts.
- Ensure that ownership and privilege levels do not change during the restore.

Virtual Backups

Virtual backups perform a full backup of the catalog store and select SAP Sybase IQ metadata, and metadata not specific to individual tables, including information specific to the freelist, backup, and checkpoint. Virtual backups do not back up table data and metadata in the IQ store. Use the **VIRTUAL** backup parameters as part of a full database backup.

Virtual backups can quickly (in seconds or minutes) back up and restore an entire database minimizing downtime. SAN technologies create multiple mirrored copies of dbfile devices, which offload maintenance tasks, such as consistency checks, on the mirrored copies. Use the **sp_iqfile** procedure to create a backup list of files that comprise the database.

You must make a separate OS-level copy of the corresponding IQ store. To restore from a virtual backup, restore the corresponding system-level copy of the IQ store, then proceed with the IQ full restore of the virtual backup.

Encapsulated Virtual Backups

In an encapsulated virtual backup, the *'shell-command'* parameter executes arbitrary shell commands as part of the backup operation:

```
BACKUP DATABASE
    FULL VIRTUAL ENCAPSULATED
      ' shell_command '
    TO archive_device
```

The **BACKUP DATABASE** command backs up the catalog and metadata; the *'shell-command'* executes an OS-level script or command that backs up the user dbspaces. Encapsulated means that the two phases are wrapped in a transactional boundary. In other words, the backup is complete and consistent–the catalog, metadata and user data all match.

Shell commands must perform correctly for system-level backups. If a shell command fails, the backup operation throws an exception. For system-level backups of table data to be consistent with the virtual backup without additional steps, the system-level backup must be made during the backup command and by the backup transaction.

This example runs the 'dd' command to copy `iqdemo`:

```
BACKUP DATABASE FULL VIRTUAL ENCAPSULATED
'dd if=iqdemo.iq of=iqdemo.iq.copy'
TO 'iqdemo.full'
```

To fully restore a database from an encapsulated virtual backup, restore the system-level backup first, then restore the virtual backup.

Decoupled Virtual Backups

A decoupled virtual backup copies all dbspaces after the backup operation is complete:

```
BACKUP DATABASE
    FULL VIRTUAL DECOUPLED
    TO archive_device
```

Decoupled virtual backups allow users to make catalog changes before the **BACKUP DATABASE** command completes the full backup. To gain that flexibility, you must perform an incremental backup after the decoupled virtual backup is complete, to resynchronized the catalog and data.

If you perform a system-level backup outside of the backup transaction, the IQ store backup will not be consistent with the IQ backup file. However, a nonvirtual IQ incremental backup, together with the virtual full backup, represents a consistent database. This is because the IQ incremental backup copies all IQ store data and metadata that have changed during or since the virtual full backup. Even the automatic commit and checkpoint that are part of the backup command modify the IQ store, making an independent system-level backup inconsistent. Trying to use the database without applying the incremental restore gives unpredictable results.

To perform a full virtual decoupled backup, first perform the full backup:

```
BACKUP DATABASE
    FULL VIRTUAL DECOUPLED
    TO 'iqdemo.full'
```

Then perform a nonvirtual incremental backup:

```
BACKUP DATABASE
    INCREMENTAL SINCE FULL
    TO 'iqdemo.isf'
```

For a fully restored database from a system-level backup with a virtual decoupled backup, restore the system-level backup, then restore of the virtual decoupled backup followed by an incremental-since-full restore.

Virtual Backups with SAN Snapshot or Shadow Hardware

Storage area network (SAN) snapshot or shadow hardware provides more flexibility in the backup process by allowing the system-level backup to take place on the shadow copy rather than on the main database. Instead of the system-level copy that is part of the virtual backup, you can create a shadow copy of the IQ store. You can then perform a system-level backup against the shadow copy. This allows the full backup to complete quickly.

See also

- *Routine Backups* on page 16
- *Cache Dbspace Backup* on page 17
- *System-Level Backups* on page 18

Previous Backups

Views, utilities, and procedures return details about previous backups.

To query **SYSIQBACKUPHISTORY** and return details about previous backups, enter:

```
select * from sysiqbackuphistory
```

In these results, each row represents a successful backup operation:

```
   bu_id bu_time                 type selective_type virtual_type dependson_id cmd                      creator  version
-------- ----------------------- ---- -------------- ------------ ------------ ------------------------ -------- -------
  312372 2013-07-15 09:13:54.000    0              0            0            0 backup database to ...   DBA            8
```

SYSIQBACKUPHISTORY is a system view that stores details about previous backup operations. See *Reference: Building Blocks, Tables, and Procedures > System Tables and Views > System Views > Alphabetical List of System Views > SYSIQBACKUPHISTORY System View*.

db_backupheader Utility

Reads the first backup archive, returns backup statistics and definitions.

Syntax:

```
db_backupheader [ path ] backup_file
```

Execute **db_backupheader** from the command line. Output includes backup statistics, database definitions, dbspace and dbfile specifics.

sp_iqbackupsummary Procedure

Summarizes backup operations.

Syntax:

```
sp_iqbackupsummary [ timestamp or backup_id ]
```

Results are similar to:

```
backup_id backup_time            backup_type  selective_type virtual_type depends_on_id creator backup_size user_comment
backup_command
--------- ---------------------  ------------ -------------- ------------ ------------- ------- ----------- ------------
-------------------
   312372 2013-07-15 09:13:54.0  Full         All inclusive  Non virtual             0 DBA          50800 (NULL)      backup
database
to ...
```

See *Reference: Building Blocks, Tables, and Procedures > System Procedures > Alphabetical List of System Stored Procedures > sp_iqbackupsummary Procedure* .

sp_iqbackupdetails Procedure

Summarizes operations performed by a particular backup.

Syntax:

```
sp_iqbackupdetails 'backup_id'
```

Results are similar to:

```
backup_id backup_time             backup_type  selective_type  depends_on_id dbspace_id dbspace_name   dbspace_rwstatus
dbspace_createid   ...
--------- ----------- ----------  ------------ --------------  ------------- ---------- ------- ------ ------------------
----------------  ...
   312372 2013-07-15 09:13:54.0   Full         All inclusive               0          0 system
ReadWrite                         0   ...
   312372 2013-07-15 09:13:54.0   Full         All inclusive               0      16384 IQ_SYSTEM_MAIN
ReadWrite                         0   ...
   312372 2013-07-15 09:13:54.0   Full         All inclusive               0      16387 iq_main
ReadWrite                         6   ...
```

This example omits some columns that appear in the output. See *Reference: Building Blocks, Tables, and Procedures > System Procedures > Alphabetical List of System Stored Procedures > sp_iqbackupdetails Procedure*.

sp_iqrestoreaction Procedure

Identifies actions required to restore the database to a consistent state for a given date.

Syntax:

```
sp_iqrestoreaction 'timestamp'
```

Results are similar to:

```
sequence_number    backup_id backup_archive_list  backup_time           virtual_type  restore_dbspace restore_dbfile
backup_comment

--------------- ------------ -------------------- --------------------- ------------- --------------- --------------
----------------------------
              1         1192  c:\\\\temp\\\\b1     2008-09-23 14:47:40.0  Non virtual
              2         1201  c:\\\\temp\\\\b2.inc 2008-09-23 14:47:40.0  Non virtual
              3         1208  c:\\\\temp\\\\b3.inc 2008-09-23 14:47:40.0  Non virtual
```

See *Reference: Building Blocks, Tables, and Procedures > System Procedures > Alphabetical List of System Stored Procedures > sp_iqrestoreaction Procedure*.

See also

- *Database Backup* on page 1
- *Database Validation* on page 3
- *Performance Options* on page 4
- *Archive Devices* on page 6
- *Queries, Utilities, and Procedures* on page 13
- *Backup Scenarios* on page 15
- *Backup Log* on page 24

Backup Log

SAP Sybase IQ logs events that occur during backups and restores to `.backup.syb`.

There is only one backup log on a server. The server must be able to read and write this file. The system administrator may want to limit access to this file by other users.

Location

Location of the backup log depends on the setting of environment variables at server startup time. If you are running more than one database server on a system, set the *$IQLOGDIR16* or *%IQLOGDIR16%* environment variable differently for each server to produce separate backup logs.

Table 5. Backup Log Location

Platform	Location
UNIX-like operating systems	The server tries to write `backup.syb` to one of the following locations, in this order: **1.** Directory specified by the *$IQLOGDIR16* environment variable **2.** Directory specified by the *$HOME* environment variable **3.** Home directory as obtained from account information **4.** *current directory* (where the server was started) The server writes `.backup.syb` to the *$HOME* directory as a hidden file, and the file name is prefixed with a ".". If the server writes the file to the *current directory*, `backup.syb` is not hidden and not prefixed.

Platform	Location
Windows	The server tries to write `backup.syb` to one of the following locations in this order: **1.** Directory specified by the *%IQLOGDIR16%* environment variable **2.** Directory that holds the server executable files.

Sample Output

The backup log contains a comma-delimited list of events that occurred during backups and restores:

```
BACKUP, 2.0, all_types.db, ASIQ, '2009-01-31 16:25:00.000', DBA,
Full, Arch, TED_FULL00, '
BACKUP, 2.0, all_types.db, ASIQ, '2009-01-31 16:53:00.000', DBA,
Incr, Arch, TED_X_bkup_inc, ''

RESTORE, 2.0, all_types.db, ASIQ, '2009-01-31 16:25:00.000', DBA,
Full, Arch, TED_FULL00, ''
RESTORE, 2.0, all_types.db, ASIQ, '2009-01-31 16:53:00.000', DBA,
Incr, Arch, TED_X_bkup_inc, ''

BACKUP, 2.0, all_types.db, ASIQ, '2009-01-31 20:07:00.000', DBA,
InSF, Arch, A_partial2_yes_sf, ''
BACKUP, 2.0, all_types.db, ASIQ, '2009-01-31 20:07:00.000', DBA,
InSF, Arch, A_partial2_yes_sf, ''
```

Maintenance

After you purge backup media, use a text editor to clean up the backup log after you purge backup media. Do not edit the backup log while a backup or restore operation is in progress. Be careful with your edits: once BACKUP DATABASE or RESTORE DATABASE writes to the backup log, it does not check its accuracy.

Note: To display only the information about a particular backup, you can run RESTORE DATABASE with the **CATALOG ONLY** option. This option displays the header file for a backup from the media rather than from the file, so that the DBA can identify what is on the tape or file.

See also

- *Database Backup* on page 1
- *Database Validation* on page 3
- *Performance Options* on page 4
- *Archive Devices* on page 6
- *Queries, Utilities, and Procedures* on page 13
- *Backup Scenarios* on page 15

- *Previous Backups* on page 22

Restoring Database Backups

The **RESTORE DATABASE** utility returns a damaged or inconsistent database to the same state as at the end of the first implicit CHECKPOINT of the backup you restored. Restore backups in the correct order, using a separate RESTORE command for each backup.

When you restore from a full backup, every block in use at the time the backup is written to disk. When you restore from an incremental backup, only the blocks that change between the previous backup (or the previous full backup) and this backup are written to disk. During an incremental restore, **RESTORE DATABASE** creates and drops dbspaces as needed to match the period of operation encompassed by the restores.

Unless specifically noted, instructions pertain to the **RESTORE DATABASE** utility that is distributed as part of SAP Sybase IQ. For third-party APIs, refer to the third-party product documentation for archive_device strings and other information.

Database Restore

Interactive SQL supports two RESTORE DATABASE syntax variations.

Syntax 1:

```
RESTORE DATABASE 'dbfile'
  'archive_device' [ FROM 'archive_device' ]…
  … [ CATALOG ONLY ]
  … [ KEY key_spec ]
  … [ [ RENAME logical-dbfile-name TO 'new-dbspace-path']...
      | VERIFY [ COMPATIBLE ] ]
```

Syntax 2:

```
RESTORE DATABASE 'database-name'
  [ restore-option ...]
  FROM 'archive_device' ...
```

For this backup statement:

```
BACKUP DATABASE READONLY DBSPACES iq_main
TO '/system1/IQ16/demo/backup/iqmain'
```

You can use either of one these **RESTORE DATABASE** commands to restore the dbspace `iq_main`:

```
//syntax 1
RESTORE DATABASE 'iqdemo.db' READONLY DBSPACES iq_main
FROM '/system1/IQ16/demo/backup/iqmain'
```

or

```
//syntax 2
RESTORE DATABASE 'iqdemo'
FROM '/system1/IQ16/demo/backup/iqmain'
```

The *dbfile* declaration in Syntax 1 refers to the location of the catalog store. The *dbfile* declaration can include a full or relative path to the location of the file.

Guidelines

- To restore an inconsistent database, or to move the database to a new location, restore from a **FULL** backup.
- To restore a database to the state before you performed any incremental backups or if your most recent backup is a **FULL** backup, restore the **FULL** backup only.
- To recover from a database failure when a **INCREMENTAL_SINCE_FULL** backup is available, restore the last **FULL** backup first, then restore the **INCREMENTAL_SINCE_FULL** backup.
- If no **INCREMENTAL_SINCE_FULL** backup is available, but you performed one or more **INCREMENTAL** backups since your last **FULL** backup, restore the **FULL** backup first, then restore the **INCREMENTAL** backups in order.

Restrictions

Depending on the type of backup you plan to restore, you may need to delete some objects and verify others.

Restore Option	Description
Full	Delete or move the catalog store (`.db`) , IQ store files (`.iq`), transaction log (`.log`), and user-defined stores. If any of these files reside in the target directory, RESTORE DATABASE generates an exception and does not restore your files.
Incremental	The catalog store must reside in the target directory. If the catalog does not exist, run a full restore to restore the catalog, then perform the incremental restore. The database must not have changed since the last restore. The catalog store and IQ store must match the files they replace on the file system. These restrictions apply to all types of incremental restores.

Database Status

In almost all cases, the database must not be running while you restore all files from a backup. This applies to **FULL**, **INCREMENTAL SINCE FULL**, or **INCREMENTAL** backups, as well as to **READWRITE FILES ONLY** backups.

- To restore a backup of read-only files, the database may or may not be running. To restore specific files in a read-only dbspace, the dbspace must be offline.

- To restore read-only files in a read-write dbspace, the dbspace can be online or offline. RESTORE DATABASE closes the read-only files, restores the files, then reopens the files.
- To restore read-only files or dbspaces, the database may be running. The read-only file pathnames not not need match the names in the backup, if they otherwise match the database system table information.

Disk Files

Specify the same number of disk files to restore the database that you used to create the backup.

Tape Sets

Position the tapes to the correct starting point before you place them in the tape device. Keep track of the order of tapes in each backup tape set, that is, the set of tapes produced in a given backup on a given archive device.

Restore the tape set that contains the backup of the catalog store first, and it must be on the first archive device. Restore all tapes in order. Do not interleave sets. Restore all the tapes in one set before you restore another. After the first set, the order in which sets are restored does not matter, as long as the order is correct within each set. Use the same number of drives to restore the tapes as you used to produce the backup, so that you do not accidentally interleave tapes from different sets.

Utility Database

All RESTORE DATABASE commands are executed from the utility database (`utility_db`), and requires exclusive access to the database.

Use the **-gd DBA** and **-gm 1** parameters to start the server. The **-gd DBA** parameter sets the privileges required to start or stop a database on a running server to users with SERVER OPERATOR privileges. The **-gm 1** parameter limits the number of concurrent connections to a single connection plus one DBA connection above the limit. This allows a user with DROP CONNECTION privileges to connect to the server and drop other connections.

1. Shutdown the database server.
2. Start a server that you can use to connect to the utility database:
   ```
   start_iq -su mypwd -gd DBA -gm 1 -n my_server
   ```
3. Start dbisql and connect to the utility database:
   ```
   dbisql -c "UID=DBA;PWD=mypwd;DBN=utility_db"
   ```
4. Run all **RESTORE DATABASE** commands from dbisql. On Windows, you do not need to redouble the backslashes in path names for tape devices for restore, as you did for BACKUP DATABASE.
5. Shutdown the server and utility database.
6. Start the database server normally.

When SAP Sybase IQ completes the restore, the database is left in the state that existed at the end of the first implicit CHECKPOINT of the backup you restored.

Database Validation

To ensure that you restored a tape set in the correct order, run **sp_iqcheckdb**. For incremental backups, run **sp_iqcheckdb** after you restore each backup. To save time, however, you can run **sp_iqcheckdb** only after you restore the last incremental backup.

See also

- *Verifying Database Backups* on page 30
- *Displaying Header Information* on page 32
- *Restoring to Raw Devices* on page 33
- *Restoring Cache Dbspaces* on page 34
- *Moving Database Files* on page 35
- *Restoring Multiplex Stores* on page 37
- *Error Recovery* on page 41

Verifying Database Backups

The **RESTORE VERIFY** and **RESTORE... VERIFY COMPATIBLE** options validate the archived backups against the database and write the results to the server log. The backup verification process can run on a host other than the database host.

RESTORE... VERIFY

RESTORE... VERIFY checks all stripes and logs the number of verified blocks to the server log:

```
I. 08/26 11:50:16. RESTORE VERIFY Started
I. 08/26 11:50:16. Total number of IQ blocks to be verified: 5944
I. 08/26 11:50:16. Total number of IQ blocks verified: 25/5944 ( 0% )
I. 08/26 11:50:16. Total number of IQ blocks verified: 5030/5944
( 84% )
I. 08/26 11:50:16. Total number of IQ blocks verified: 5944/5944
( 100% )
I. 08/26 11:50:16. RESTORE VERIFY Successfully Complete
```

If you specify **RESTORE... VERIFY** for an incremental restore, SAP Sybase IQ does not look for any dbspaces or perform compatibility checks. No warning is reported, even if the files do not exist. Compatibility checks are performed only with **RESTORE… VERIFY COMPATIBLE**.

RESTORE… VERIFY COMPATIBLE

RESTORE… VERIFY COMPATIBLE checks the compatibility of an incremental archive with the current database. If the database files do not exist, **RESTORE…VERIFY COMPATIBLE** logs an exception. **RESTORE…VERIFY COMPATIBLE** opens the dbspaces in read-only mode to perform consistency checking. No dbspaces are modified. If the catalog store or any dbspaces are missing, **RESTORE…VERIFY COMPATIBLE** throws an error and the operation fails.

If you specify **RESTORE...VERIFY COMPATIBLE** for a full backup, the **COMPATIBLE** keyword is ignored; no compatibility checks are necessary to restore a full backup.

In incremental restores, if the database has been modified or the particular incremental archive is not the correct archive for the database, **RESTORE VERIFY COMPATIBLE** reports the error `Database has changed since last restore` or `This restore cannot immediately follow the previous restore`.

Verification Error Reporting

In most cases, an exception terminates the verification process. If verification encounters these errors, the process continues to check the archive and logs information for the errors detected.

The errors for which verification can continue are:

- `Header of block to be restored appears to be corrupted. (SQLCODE -10120111, SQLSTATE QUA11)`
- `Media data appears corrupted (bad checksum). (SQLCODE -1012012, SQLSTATE QUA12)`
- `Media meta data appears corrupted (boundary record). (SQLCODE -1012013, SQLSTATE QUA13)`
- `Media meta data appear corrupted (multiple begin boundary records). (SQLCODE -1012014, SQLSTATE QUA14`

If any of these errors are found and the verification process can continue to the end of the archive, SAP Sybase IQ reports this error: `The verification of the provided archive has failed. Please check the server log for details of the errors thrown during verify.`

If any error pertaining to **RESTORE** is found other than the errors above, the error that occurred is reported, and the verification process stops.

Note: The verification of a backup archive is different than the database consistency checker (DBCC) verify mode (`sp_iqcheckdb 'verify...'`). **RESTORE VERIFY** validates the consistency of the backup archive to be sure it can be restored, whereas DBCC validates the consistency of the database data.

Run `sp_iqcheckdb 'verify...'` before taking a backup. If an inconsistent database is backed up, then restored from the same backup archive, the data continues to be in an inconsistent state, even if **RESTORE VERIFY** reports a successful validation.

See also

- *Database Restore* on page 27
- *Displaying Header Information* on page 32
- *Restoring to Raw Devices* on page 33
- *Restoring Cache Dbspaces* on page 34

- *Moving Database Files* on page 35
- *Restoring Multiplex Stores* on page 37
- *Error Recovery* on page 41
- *RESTORE DATABASE Statement* on page 73

Displaying Header Information

The **RESTORE... CATALOG ONLY** option reads only the backup header from the archive, and writes the results to the backup log in the same format as the log entry. **RESTORE... CATALOG ONLY** does not restore any data, either from the catalog store or the IQ store.

Include *database-name* and *archive_device* as part of the command. Omit any additional clauses:

```
RESTORE DATABASE 'database-name'
    FROM 'archive_device'
    CATALOG ONLY
```

This example reads the `iqdemo` archive:

```
RESTORE DATABASE 'iqdemo.db'
    FROM '/disk1/users/jones/backup/iqdemo'
    CATALOG ONLY
```

and writes this result to the backup log:

```
RESTORE, -1988637423.0, bigendian_420111.db,
        ASIQ, '2013-01-11
       06:57:00.000', DBA, Full, Arch, bigendian_420111_backup, ''
BACKUP, 453495113.0, iqdemo.db, ASIQ, '2013-01-23 10:33:00.000',
DBA,
        Full, Arch, /disk1/users/jones/backupiqdemo, ''
RESTORE, 453496081.0, ,  ASIQ , '2013-01-23 10:33:00.000', DBA,
        Full, Arch, /disk1/users/jones/backup/iqdemo,
      ''
```

Note: You can get more information about the backup archive using the command line utility **db_backupheader**, which accepts the file path corresponding to the first backup archive. The utility reads the backup archive file. It does not connect to the database.

See also

- *Database Restore* on page 27
- *Verifying Database Backups* on page 30
- *Restoring to Raw Devices* on page 33
- *Restoring Cache Dbspaces* on page 34
- *Moving Database Files* on page 35
- *Restoring Multiplex Stores* on page 37
- *Error Recovery* on page 41

Restoring to Raw Devices

To restore a backup to a raw device, the device must be large enough to hold both the dbspace and the space reserved for the operating system.

Query the system tables to determine whether the raw device is large enough:

```
SELECT segment_type, file_name, block_count,
data_offset, block_size,
(block_count * block_size) + data_offset AS raw_size
FROM SYS.SYSIQFILE, SYS.SYSIQINFO
where segment_type !='Msg' ORDER BY 1,2
```

The query returns results similar to this:

```
segment_type  file_name         block_count  data_offset block_size
raw_size
Main          iqdemo.iq          12800        65536       8192
104923136
Main          iqdemo_main.iq      2800        65536       8192
104923136
Temp          iqdemo.iqtmp        3200        65536       8192
26279936
```

Columns returned by the query:

Column Name	Description
`segment_type`	`Main` or `Temp` segments, but not message files (type `Msg`).
`file_name`	Dbspace name.
`block_count`	Number of blocks in use.
`data_offset`	Number of bytes reserved for the operating system.
`block_size`	Number of bytes per block.
`raw_size`	Minimum raw device size, in bytes required to restore the dbspace. The target device must be at least 10MB larger than the original raw device.

Example

In this scenario, the database `iquser` resides on a raw device. This command performs a FULL database backup to two tape devices:

```
BACKUP DATABASE
TO '/dev/rmt/0n'
TO '/dev/rmt/1n'
WITH COMMENT 'Jan 18 full backup of iquser'
```

The catalog store is backed up first, to `/dev/rmt/0n`. The IQ store is backed up next, to both tapes.

Assume that a media failure occurs and the raw partition is now unusable. To restore a user-defined dbfile named `IQ_USER` to a new raw partition, `/ dev/rdsk/c1t5d2s1`:

```
RESTORE DATABASE 'iquser'
FROM '/dev/rmt/0n'
FROM '/dev/rmt/1n'
RENAME IQ_SYSTEM_MAIN TO '/dev/rdsk/c2t0d1s1'
RENAME IQ_SYSTEM_TEMP TO '/dev/rdsk/c2t1d1s1'
RENAME IQ_SYSTEM_MSG TO 'iquser.iqmsg'
RENAME IQ_USER TO '/dev/rdsk/c1t5d2s1'
```

You can also issue these commands using only the last **RENAME** clause, since only one dbspace is being restored to a new location. Listing all of the files or raw partitions, as shown here, ensures that you know exactly where each will be restored.

See also

- *Database Restore* on page 27
- *Verifying Database Backups* on page 30
- *Displaying Header Information* on page 32
- *Restoring Cache Dbspaces* on page 34
- *Moving Database Files* on page 35
- *Restoring Multiplex Stores* on page 37
- *Error Recovery* on page 41

Restoring Cache Dbspaces

If your installation uses the cache dbspace for direct-attached storage, preserve any files that were added to the cache dbspace before restoring the database.

Multiplex Considerations

To restore to the same location, make sure the cache dbspaces files and devices for all secondary nodes exist as before. If a file is not physically present in the secondary node path, you must drop the file or dbspace after the database is restored.

To restore a multiplex with a new location path for the database and dbspaces, you must either:

- Use the DROP MULTIPLEX SERVER statement to drop all secondary nodes; then create the cache dbspace on each node; then add files manually, or
- Use the ALTER MULTIPLEX SERVER statement to point each secondary node to the path of its cache dbspace and dbfiles.

See also

- *Database Restore* on page 27
- *Verifying Database Backups* on page 30
- *Displaying Header Information* on page 32

Moving Database Files

Redirect the catalog, or use the **RENAME** clause to move the database to a new location on the file system.

Redirect the Catalog Store

To move the catalog store (*db_file*) to a new location on the file system, redirect the catalog to the target directory with a new *db_file* file name:

```
RESTORE DATABASE 'new-file-path''new-db_file-name'
  FROM  'archive_device '
  ...
```

RESTORE copies the dbfiles to the target location, and renames the catalog (*db_file*) to *new-db_file-name*`.db`. For example, to redirect a backup of `iqdemo.db` to a new location on the file system, use:

```
RESTORE DATABASE 'c:\\newdir\\iqnew.db'
 FROM 'c:\\iq\\backup1'
 FROM 'c:\\iq\\backup2'
```

Contents of the `newdir` directory:

```
iqdemo.iq
iqdemo.iqmsg
iqdemo.iqtmp
iqdemo.log
iqdemo_main.iq
iqnew.db
```

RESTORE renamed the backed up catalog (`iqdemo.db`) to `iqnew.db`. All other dbfile names remain unchanged.

RENAME Clause

Use the RENAME clause to move one or more database files to a new location:

```
RESTORE DATABASE 'new-database-name'
 'FROM  'archive_device '
 RENAME file-name TO new-file-path
 ...
```

Specify each dbfile file_name as it appears in the SYSIQDBFILE table. Specify the new-dbspace-path as the new raw partition, full, or relative path for that dbspace.

Do not use the RENAME clause to move the SYSTEM dbspace, which holds the catalog store. To move the catalog store, and any relative files not specified in a RENAME clause, specify a new location as part of the *new-database-name* parameter.

This example moves a user-defined dbfile (`iquser`) on a raw partition to a new raw partition (`/dev/rdsk/c1t5d2s1`). No other database files are affected. The first code block restores the full backup:

```
RESTORE DATABASE 'iquser'
 FROM '/dev/rmt/0n'
 FROM '/dev/rmt/1n'
 RENAME IQ_SYSTEM_MAIN TO '/dev/rdsk/c2t0d1s1'
 RENAME IQ_SYSTEM_TEMP TO '/dev/rdsk/c2t1d1s1'
 RENAME IQ_SYSTEM_MSG TO 'iquser.iqmsg'
 RENAME IQ_USER TO '/dev/rdsk/c1t5d2s1'
```

The second code block restores the incremental backup:

```
RESTORE DATABASE 'iquser'
 FROM '/dev/rmt/0n'
 RENAME IQ_SYSTEM_MAIN TO '/dev/rdsk/c2t0d1s1'
 RENAME IQ_SYSTEM_TEMP TO '/dev/rdsk/c2t1d1s1'
 RENAME IQ_SYSTEM_MSG TO 'iquser.iqmsg'
 RENAME IQ_USER TO '/dev/rdsk/c1t5d2s1'
```

In this example, you can also issue these commands using only the last RENAME clause, since only one dbspace is being restored to a new location. Listing all files or raw partitions, as shown here, ensures that you know exactly where each will be restored.

Note: When you move a database, you may need to modify your data sources, configuration files, and integrated logins to reflect the new location.

Transaction Log

When you move a database, you can rename all files except the transaction log. SAP Sybase IQ continues to write to the old log file name, in the location where the catalog store file (the `.db` file) is after the database is restored.

Use **dblog** to move or rename the `.log` file:

```
dblog [options] database-file
```

Before issuing the command, stop the server. After you rename the log, retain the old log until the next backup, in case it is needed for recovery from a media failure.

See *Utility Guide > dblog Database Administration Utility*.

See also

- *Database Restore* on page 27
- *Verifying Database Backups* on page 30
- *Displaying Header Information* on page 32
- *Restoring to Raw Devices* on page 33
- *Restoring Cache Dbspaces* on page 34

- *Restoring Multiplex Stores* on page 37
- *Error Recovery* on page 41

Restoring Multiplex Stores

Run **RESTORE DATABASE** commands only on the coordinator node. You cannot run RESTORE operations against a secondary server. Before you restore, contact Technical Support to verify that a restore operation is necessary.

You need never restore a coordinator node due to secondary node problems. If you cannot open your database on a secondary server, synchronize the server.

See also

- *Database Restore* on page 27
- *Verifying Database Backups* on page 30
- *Displaying Header Information* on page 32
- *Restoring to Raw Devices* on page 33
- *Restoring Cache Dbspaces* on page 34
- *Moving Database Files* on page 35
- *Error Recovery* on page 41

Restoring Multiplex Stores to a Different Location

Restore operations vary, depending on the location from which you restore.

Prerequisites

- Confirm that there are database home directories for each server. If there are not, create them or restore them from file system backups.
- If this is not the first time you have restored to the new location, shut down all multiplex servers running at the destination location (coordinator and secondary servers). The multiplex at the location from which where the backup was taken may continue running.

  Note: If automatic startup is enabled in your ODBC configuration, there may be users on the same machine as the server who are set up to automatically start the server. Prevent this from happening while you are restoring the database.
- Confirm that the database shut down successfully:

Platform	Actions
UNIX	`% ps -ef | grep iqsrv16` If you see an active **iqsrv16** process with name of a multiplex, stop the process.

Platform	Actions
Windows	In Task Manager, check the Processes tab for `iqsrv16.exe`, or find the IQ Server icon in the system tray and select **Shutdown**.

- Make a file system copy of the `.iqmsg` file. If you have message log archiving configured, see *Back Up the Right Files*.

Task

1. Shut down the original coordinator, then start the utility database from the coordinator server directory using the coordinator server's name:

```
% start_iq -n coordinator_svr -c 32MB
-x tcpip(port=1234)
```

2. Connect to the utility database (`utility_db`):

```
% dbisql -c "eng=coordinator_svr;uid=DBA;pwd=SQL;
dbn=utility_db" -host myhost -port 1234
```

3. Run the RESTORE command with a new location path for the database and its dbspaces. To restore certain dbspace files to a different path, specify a `RENAME` clause. Perform full and any incremental restore operations in sequence, without stopping the utility database.

   Warning! Stopping the utility database between full and incremental restore operations may invalidate the catalog and render the restored database unusable.

4. Start the restored database either by reconnecting to the `utility_db` server and specifying the restored database file name, or by stopping the server and restarting it with the restored database. If you restart the server, use the single-node and override flags (**-iqmpx_sn 1 -iqmpx_ov 1**).
5. Use DROP MULTIPLEX SERVER to drop all the secondary nodes. For example:

```
DROP MULTIPLEX SERVER node_w3_skm
```

   Once you drop the last secondary node, the coordinator shuts down automatically, signifying conversion to simplex.
6. Restart the coordinator without the single-node or override switch.
7. Re-create all secondary nodes with the correct location path, including the database file extension (.DB):

```
CREATE MULTIPLEX SERVER node_r2_skm DATABASE
'/sunx3005/mpx_simdb.db'
HOST 'localhost' PORT 8998
ROLE READER STATUS INCLUDED
```

   After you create the first secondary node, the server automatically shuts down, signifying conversion to multiplex.
8. When you restart the coordinator, you see a warning in the server log about the multiplex environment being invalid. This warning is generated if IQ_SYSTEM_TEMP dbspace

does not contain any files, and is the case for all the secondary nodes you created in step 7. Ignore this warning for now.

9. If cache dbspaces existed on one or more secondary nodes, re-create the cache dbspace and dbfiles on those secondary nodes. The cache dbspace and dbfiles were not automatically restored when you dropped and re-created secondary nodes in step 7.
   - Use the DROP MULTIPLEX SERVER statement to drop all secondary nodes; then create the cache dbspace on each node; then add files manually, or
   - Use the ALTER MULTIPLEX SERVER statement to point each secondary node to the path of its cache dbspace and dbfiles.
10. Synchronize and restart the secondary servers.
11. Connect to each secondary server and add files in IQ_SYSTEM_TEMP.
12. Run **sp_iqmpxvalidate** on the coordinator. It should report `no error detected`.

To restore an exact copy of the multiplex to a different location, when copies of all of the server's temporary files exist at the new location, replace steps 5 through 12 with:

Use ALTER MULTIPLEX SERVER to alter the server name, host, port, and database path of each server.

See also

- *Restoring Multiplex Stores to the Same Location* on page 39
- *Restoring Read-only Backups for a Coordinator* on page 41
- *Restoring Cache Dbspaces* on page 34

Restoring Multiplex Stores to the Same Location

Restore operations vary depending on where you are restoring the data.

1. Confirm that database home directories for each server still exist. If they do not, create them or restore them from file system backups.
2. Shut down every server in the multiplex (coordinator and all secondary servers).

   Note: If automatic startup is enabled in your ODBC configuration, users on the same machine as the server may be set up to start the server automatically. Prevent this from happening while you are restoring the database.
3. Confirm that the database shut down successfully:

Platform	Actions
UNIX	`% ps -ef | grep iqsrv16` If you see an active **iqsrv16** process with name of a multiplex, stop the process.

Platform	Actions
Windows	IIn Task Manager, check the Processes tab for `iqsrv16.exe`, or find the IQ Server icon in the system tray and select **Shutdown**.

4. Move files required for debugging and reconfiguring the multiplex.
   - Make a file system copy of the `.iqmsg` file. If you have message log archiving configured, see *Back Up the Right Files*.
   - On each server, preserve any files that were added to IQ_SYSTEM_TEMP for that server. These files are of the form `dbname.iqtmp` if you used an operating system file, or they may be raw devices. If the IQ temporary store is damaged, start the server with the **-iqnotemp** switch to drop and re-create the temporary store dbspaces. For more information, see the *Release Bulletin*.

     Either drop the database, or delete the following files from the coordinator:

     ```
     <database_home>/<dbname>.db
     ```

     ```
     <database_home>/<dbname>.log
     ```

     If a query server is damaged, however, drop it and re-create it after **RESTORE**. Then follow the instructions in *Restoring Multiplex Stores to a Different Location*.
   - If you have cache dbpaces on secondary nodes, preserve all cache dbspace files on those nodes. See *Restoring Cache Dbspace Restore*.
5. Start the utility database from the coordinator server directory. Use any valid identifier as the server name except the name of a registered secondary server. If you use the coordinator name, rename the coordinator after the restore.

   ```
   % start_iq -n utility_startup_svr -c 32m
   -x 'tcpip{port=1234}'
   ```
6. Connect to the utility database (`utility_db`):

   ```
   % dbisql -c "eng=utility_startup_svr;uid=DBA;pwd=SQL;
   dbn=utility_db"
   ```
7. Run the **RESTORE** command. To restore certain dbspace files to a different path, specify a **RENAME** clause. For details, see *RESTORE Statement* in *Reference: Statements and Options*.
8. Shut down the utility database.
9. Make sure that the temporary dbspaces exist as before, on raw devices or as files of the correct length. For information about starting the server without using the IQ temporary store, see the *Release Bulletin* for your platform.
10. Start the coordinator server and, if restoring to the same location, synchronize the secondary servers.
11. Start the secondary servers.

See also

- *Restoring Multiplex Stores to a Different Location* on page 37

- *Restoring Read-only Backups for a Coordinator* on page 41
- *Restoring Cache Dbspaces* on page 34
- *Back Up the Correct Files* on page 19

Restoring Read-only Backups for a Coordinator

Restore a coordinator without renaming the utility database to use the coordinator's name. This is the only supported read-only selective restore method for a multiplex coordinator.

Use this procedure to correct problems resulting from inadvertently restoring read-only dbspaces from a read-write archive, or vice versa.

1. Start the utility server with any server name except that of a secondary node.
2. Connect to the utility_db and run the **RESTORE** statement for the read-write database. Use **RENAME** clauses to move dbfiles to the corresponding locations.
3. Disconnect and stop the utility server.
4. Start the restored database. If the database has been moved to a different location, start the server with **-iqmpx_sn 1**and **-iqmpx_ov 1** flags.
5. Run **ALTER DBSPACE <dbspace name>** offline for the read-only dbspaces that have been backed up on the separate read-only backup only.
6. Disconnect and stop the server.
7. Start the utility database with any server name except that of a secondary node.
8. Connect to the utility server and run the restore command for the read-only dbspace.

Next

You may restore databases either completely or selectively (by restoring only read-write dbspaces, or a set of read-only dbspaces or read-only files).

See also

- *Restoring Multiplex Stores to a Different Location* on page 37
- *Restoring Multiplex Stores to the Same Location* on page 39

Error Recovery

Solutions to common RESTORE errors.

- If an incremental restore fails early in the operation, the database is still usable (assuming it existed and was consistent before the restore).
- RESTORE assumes an invalid tape device name refers to a disk file and tries to read from it.
- If a full restore fails, you do not have a usable database.
- If a failure occurs after a certain point in the operation, the restore program marks the database as inconsistent. In this case recovery is only possible by means of a **FULL**

RESTORE. If you were performing a **FULL RESTORE** when the failure occurred, you may need to go back to the previous **FULL BACKUP**.

Reconnecting After You Restore

SAP Sybase IQ requires the **DBF** parameter and database file name to connect to a database when you use Interactive SQL and restore the database while connected to `utility_db`.

Including the **DBF** parameter:

```
CONNECT USING 'uid=DBA;pwd=sql;dbf=node1/users/localhost/mydb.db;
links=tcpip{host=serv1;port=1234};eng=serv1_iqdemo'
```

Returns:

```
CONNECT DATABASE mydb USER DBA IDENTIFIED BY SQL
```

To avoid this error, enter a **START DATABASE** command while connected to `utility_db`:

```
START DATABASE mydb
```

Use this method when connecting via Interactive SQL.

See also

- *Database Restore* on page 27
- *Verifying Database Backups* on page 30
- *Displaying Header Information* on page 32
- *Restoring to Raw Devices* on page 33
- *Restoring Cache Dbspaces* on page 34
- *Moving Database Files* on page 35
- *Restoring Multiplex Stores* on page 37

System Recovery and Database Repair

Learn about normal SAP Sybase IQ server recovery, special recovery modes, how to verify database consistency, and how to repair database inconsistencies.

When you restart the database server, SAP Sybase IQ attempts to recover automatically. If the server cannot recover and restart, especially after a system failure or power outage, the database may be inconsistent.

Recovery and Repair Overview

If your SAP Sybase IQ server or database has problems restarting, use this information to diagnose database startup problems, verify the consistency of databases, and repair databases.

If you can restart the server after a failure, verify your database using the **sp_iqcheckdb** stored procedure, preferably before allowing users to connect

If you have trouble starting a server or database, if the database starts but users are unable to connect to it, or if problems are found during database verification, you may need to perform a forced recovery, restore the database, recover leaked space, or repair indexes.

Examining the Server Log and IQ Message Log

To determine what type of recovery or repair is needed, you need information from your server log (`servername.nnnn.srvlog`) and IQ message log (`dbname.iqmsg`). Be sure to retain this information so you can provide it to Sybase Technical Support if necessary.

For example, if data inconsistency is detected, the *dbname.iqmsg* file may include detailed diagnostic information.

Normal Recovery

During system recovery, any uncommitted transactions are rolled back and any disk space used for old versions (snapshots of database pages that were being used by transactions that did not commit) returns to the pool of available space.

After normal recovery, the database then contains only the most recently committed version of each permanent table, unless it is a multiplex database. A multiplex database contains all versions accessible to secondary servers.

During recovery from a system failure or normal system shutdown, SAP Sybase IQ reopens all connections that were active. If the **-gm** option, which sets the number of user connections, was in effect at the time of the failure, you need to restart the SAP Sybase IQ server with at least as many connections as were actually in use when the server stopped.

Database Verification

Use **sp_iqcheckdb** for database verification.

Check the consistency of your database as soon as possible after the server restarts following an abnormal termination, such as a power failure, and before performing a backup of the database.

You can use the **sp_iqcheckdb** stored procedure to detect and repair database consistency problems.

The sp_iqcheckdb Stored Procedure

The SAP Sybase IQ Database Consistency Checker (DBCC) performs database verification. The **sp_iqcheckdb** stored procedure, in conjunction with server startup options, is the interface to DBCC.

You select the different modes of check and repair by specifying an **sp_iqcheckdb** command string. **sp_iqcheckdb** reads every database page and checks the consistency of the database, unless you specify otherwise in the command string.

Note: On a secondary server **sp_iqcheckdb** does not check the free list. It performs all other checks.

DBCC has three modes that perform increasing amounts of consistency checking and a mode for resetting allocation maps. Each mode checks all database objects, unless individual dbspaces, tables, partitions, indexes, or index types are specified in the **sp_iqcheckdb** command string. If you specify individual table names, all indexes within those tables are also checked.

Note: The **sp_iqcheckdb** stored procedure does not check referential integrity or repair referential integrity violations.

DBCC Performance

The execution time of DBCC varies according to the size of the database for an entire database check, the number of tables or indexes specified, and the size of the machine. Checking only a subset of the database, i.e., only specified tables, indexes, or index types, requires less time than checking an entire database.

For the best DBCC performance, be as specific as possible in the **sp_iqcheckdb** command string. Use the 'allocation' or 'check' verification mode when possible and specify the names of tables or indexes, if you know exactly which database objects require checking.

sp_iqcheckdb Check Mode

In check mode, **sp_iqcheckdb** performs an internal consistency check on all IQ indexes and checks that each database block has been allocated correctly. All available database statistics

are reported. This mode reads all data pages and can detect all types of allocation problems and most types of index inconsistencies. Check mode should run considerably faster than verify mode for most databases.

When to run in check mode, if metadata, null count, or distinct count errors are returned when running a query.

Examples of check mode:

Table 6. sp_iqcheckdb Check Mode Examples

Command	Description
`sp_iqcheckdb 'check database'`	Internal checking of all tables and indexes in the database
`sp_iqcheckdb 'check table t1'`	Default checking of all indexes in table `t1`
`sp_iqcheckdb 'check index t1c1hg'`	Internal checking of index `t1c1hg`
`sp_iqcheckdb 'check indextype FP database'`	Checking of all indexes of type FP in the database

sp_iqcheckdb Verify Mode

In verify mode, **sp_iqcheckdb** performs an intra-index consistency check, in addition to internal index consistency and allocation checking. All available database statistics are reported. The contents of each non-FP index is verified against its corresponding FP index(es). Verify mode reads all data pages and can detect all types of allocation problems and all types of index inconsistencies.

When to run in verify mode, if metadata, null count, or distinct count errors are returned when running a query.

Examples of verify mode:

Table 7. sp_iqcheckdb Verify Mode Examples

Command	Description
`sp_iqcheckdb 'verify database'`	Verify contents of all indexes in the database
`sp_iqcheckdb 'verify table t1'`	Verify contents of all indexes in table `t1`
`sp_iqcheckdb 'verify index t1c1hg'`	Verify contents of index `t1c1hg`
`sp_iqcheckdb 'verify indextype HG table t1'`	Verify contents of all HG indexes in table `t1`

Note: If you check individual non-FP indexes in check mode, the corresponding FP index(es) are automatically verified with internal consistency checks and appear in the DBCC results.

sp_iqcheckdb Allocation Mode

In allocation mode, **sp_iqcheckdb** checks that each database block is allocated correctly according to the internal physical page mapping structures (blockmaps). Database statistics pertaining to allocation are also reported. This mode executes very quickly. Allocation mode, however, does not check index consistency and cannot detect all types of allocation problems.

When to run in allocation mode:

- To check for leaked blocks or inconsistent indexes due to multiply owned blocks
- After forced recovery, run **sp_iqcheckdb** in dropleaks mode to reset the allocation map (must use database as the target)
- To check for duplicate or unowned blocks (use database or specific tables or indexes as the target)
- If you encounter page header errors

Examples of allocation mode:

Table 8. sp_iqcheckdb Allocation Mode Examples

Command	Description
`sp_iqcheckdb 'allocation database'`	Allocation checking of entire database
`sp_iqcheckdb 'allocation database dumpleaks'`	Allocation checking of entire database and print block numbers for leaked blocks to IQ message file
`sp_iqcheckdb 'allocation table t1'`	Allocation checking of table `t1`
`sp_iqcheckdb 'allocation index t1c1hg'`	Allocation checking of index `t1c1hg`
`sp_iqcheckdb 'allocation indextype LF table t2'`	Allocation checking of all LF indexes in table `t2`

If some partitions of the table are offline, you can specify a partition target to check only part of a table.

You can combine all modes and run multiple checks on a database in a single session. In the following example, **sp_iqcheckdb** performs a quick check of partition `p1` in table `t2`, a detailed check of index `i1`, and allocation checking for the entire database using half of the CPUs:

```
sp_iqcheckdb 'check table t2 partition p1
verify index i1
allocation database resources 50'
```

Allocation mode options are only allowed with the DBCC command 'allocation database'.

The following allocation mode options print block numbers for affected database blocks to the IQ message file:

- **dumpleaks –** leaked blocks
- **dumpdups –** duplicate blocks
- **dumpunallocs –** unallocated blocks

The resetclocks option corrects the values of internal database versioning clocks, in the event that these clocks are slow. Do not use the resetclocks option for any other purpose unless you contact Technical Support.

The resetclocks option must be run in single user mode and is only allowed with the DBCC command 'allocation database'. The syntax of the resetclocks command is:

```
sp_iqcheckdb 'allocation database resetclocks'
```

sp_iqcheckdb Dropleaks Mode

When the SAP Sybase IQ server runs in single-node mode, you can use dropleaks mode with either a database or dbspace target to reset the allocation map for the entire database or specified dbspace targets. If the target is a dbspace, then the dropleaks operation must also prevent read-write operations on the named dbspace. All dbspaces in the database or dbspace list must be online.

In the following example, the first statement resets allocation maps for the entire database, and the second statement resets allocation maps for the dbspace `dbsp1`.

```
sp_iqcheckdb 'dropleaks database'
sp_iqcheckdb 'dropleaks dbspace dbsp1'
```

Note: Use **sp_iqrebuildindex** to repair index errors.

See also

- *sp_iqcheckdb Procedure* on page 80

sp_iqcheckdb Output

The output of **sp_iqcheckdb** consists of an extensive list of statistics and any errors reported by DBCC.

Only non-zero values are displayed. Lines containing errors are flagged with asterisks (*****). Note that if you encounter errors, some of the statistics reported by DBCC may be inaccurate.

The output of **sp_iqcheckdb** is always copied to the IQ message file (`.iqmsg`). To redirect the **sp_iqcheckdb** output to a file, enter the following command:

```
sp_iqcheckdb >#  file_name
```

where *file_name* is the name of the file to receive the output.

When the **DBCC_LOG_PROGRESS** option is ON, **sp_iqcheckdb** sends progress messages to the IQ message file. These messages allow the user to follow the progress of the **sp_iqcheckdb** procedure as it executes.

The following is sample progress log output of the command `sp_iqcheckdb 'check database'`

```
IQ Utility Check Database
Start CHECK STATISTICS table: tloansf
Start CHECK STATISTICS for field: aqsn_dt
Start CHECK STATISTICS processing index:
ASIQ_IDX_T444_C1_FP
Start CHECK STATISTICS processing index:
tloansf_aqsn_dt_HNG
Done CHECK STATISTICS field: aqsn_dt
```

Future Version Errors

If you see the message `DBCC Future Version Errors`, a DDL operation has been performed since the DBCC transaction began. DBCC continues to process the remaining tables, but leaked block checking is not performed and statistics do not include the tables that were skipped.

To avoid DBCC Future Version errors, execute the **COMMIT** command before you run **sp_iqcheckdb**.

The following DBCC output indicates a Future Version error:

```
==============================|====================|=====
DBCC Verify Mode Report       |                    |
==============================|====================|=====
** DBCC Future Version Errors |1                   |*****
```

Sample Output of Valid Database

The following is an example of running **sp_iqcheckdb** in verify mode. No errors are detected, there is no leaked space, the database allocation is consistent, and all indexes are consistent.

The command line for this example is **sp_iqcheckdb 'verify database'**. Note that DBCC verifies all indexes, but the index verification output shown here is abbreviated.

Each index that DBCC determines to be consistent is marked as verified in the result set.

```
      Stat                        Value                          Flags
=======================|==================================|=====
DBCC Verify Mode Report |                                  |
=======================|==================================|=====
   DBCC Status          |No Errors Detected                |
   DBCC Work units      |                                  |
   Dispatched           |75                                |
   DBCC Work units      |                                  |
   Completed            |75                                |
=======================|==================================|=====
Index Summary           |                                  |
=======================|==================================|=====
```

```
   Verified Index Count |86                                  |
========================|==================================|=====
Allocation Summary      |                                  |
========================|==================================|=====
   Blocks Total         |8192                              |
   Blocks in Current    |                                  |
   Version              |4855
 |
   Blocks in All        |                                  |
   Versions             |4855                              |
   Blocks in Use        |4855                              |
   % Blocks in Use      |59                                |
========================|==================================|=====
Allocation Statistics   |                                  |
========================|==================================|=====
   DB Extent Count      |1                                 |
   Blocks Created in    |                                  |
   Current TXN          |211                               |
   Blocks To Drop in    |                                  |
   Current TXN          |212                               |
   Marked Logical       |                                  |
   Blocks               |8240                              |
   Marked Physical      |                                  |
   Blocks               |4855                              |
   Marked Pages         |515                               |
   Blocks in Freelist   |126422                            |
   Imaginary Blocks     |121567                            |
   Highest PBN in Use   |5473                              |
   Total Free Blocks    |3337                              |
   Usable Free Blocks   |3223                              |
   % Total Space        |                                  |
         Fragmented     |1                                 |
   % Free Space         |                                  |
         Fragmented     |3                                 |
   Max Blocks Per Page  |16                                |
   1  Block Page Count  |104                               |
   3  Block Page Count  |153                               |
   ...                  |                                  |
   16 Block Hole Count  |199                               |
========================|==================================|=====
Index Statistics        |                                  |
========================|==================================|=====
   ...
   Verified Index       |fin_data.DBA.ASIQ_IDX_T209_C3_HG  |
   Verified Index       |fin_data.DBA.ASIQ_IDX_T209_C4_FP  |
   Verified Index       |product.DBA.ASIQ_IDX_T210_C1_FP   |
   ...
   Verified Index       |employee.DBA.ASIQ_IDX_T212_C20_FP |
   Verified Index       |iq_dummy.DBA.ASIQ_IDX_T213_C1_FP  |
   FP Indexes Checked   |68                                |
   HNG Indexes Checked  |1                                 |
   HG Indexes Checked   |17                                |
========================|==================================|=====
   ...
```

The DBCC output also contains extensive statistical information grouped under headings such as Container Statistics, Buffer Manager Statistics, catalog Statistics, Connection Statistics, and Compression Statistics. You can see an example of the available statistics by executing the command `sp_iqcheckdb 'verify database'` after connecting to the SAP Sybase IQ demonstration database `iqdemo`.

Resource Issues Running sp_iqcheckdb

sp_iqcheckdb reports resource issues encountered while executing.

Messages describing resource issues are reported in the **sp_iqcheckdb** output or in the `.iqmsg` file.

- `Out of memory` and `DBCC Out of Memory Errors` You do not have enough memory for this operation. You may need to prevent other IQ operations or other applications from running concurrently with the **sp_iqcheckdb** stored procedure.
- `No buffers available` and `DBCC Out of Buffers Errors` The DBA may need to increase the buffer cache size.

Buffer cache sizes are set permanently using the database option `TEMP_CACHE_MEMORY_MB`. Use the server startup switches **-iqmc** and **-iqtc** to override the buffer cache size values set using the database options.

Do not run multiple database consistency checks at the same time, as DBCC is optimized to run one instance.

The CPU utilization of DBCC can be limited by specifying the **sp_iqcheckdb** parameter **resources resource-percent**, which controls the number of threads with respect to the number of CPUs. The default value of *resource-percent* is 100, which creates one thread per CPU and should match the load capacity of most machines. Set *resource-percent* to a value less than 100 to reduce the number of threads, if you are running DBCC as a background process. The minimum number of threads is 1.

If *resource-percent* > 100, then there are more threads than CPUs, which may increase performance for some machine configurations.

The database option **DBCC_PINNABLE_CACHE_PERCENT** can be used to tune DBCC buffer usage. The default of **DBCC_PINNABLE_CACHE_PERCENT** is to use 50% of cache. See *Reference: Statements and Options*.

Database Repair

Allocation problems can be repaired by running **sp_iqcheckdb** in **dropleaks** mode.

If DBCC detects index inconsistencies while attempting allocation repair, an error is generated and allocation problems are not fixed.

Analysis of Index Errors

Use **sp_iqcheckdb** to analyze index inconsistencies.

Sample of Output with Inconsistent Index

The following is an example of the type of output you see when you run **sp_iqcheckdb** and there is index inconsistency. DBCC displays both a summary and details about the indexes checked. The Index Summary section at the top of the report indicates if any inconsistent indexes were found. The names of the inconsistent indexes and the type(s) of problems can be found in the index statistics section. The lines with asterisks (*****) contain information about inconsistent indexes.

Extra, missing, or duplicate RID errors are the most common types of errors reported. These errors are an indication that the index is misrepresentative of the data and may give incorrect results or cause other failures. These errors are generally accompanied by other errors indicating the specifics of the inconsistencies.

In this example, DBCC reports an inconsistent HNG index. Because the corresponding FP index checks are good, the FP index can be used with **sp_iqrebuildindex** to repair the damaged HNG index.

The command line executed for this example is `sp_iqcheckdb 'verify database'`. Note that DBCC produces a detailed report, but some lines of the output have been removed in this example.

```
        Stat                          Value                      Flags
========================|==================================|=====
DBCC Verify Mode Report |                                  |
========================|==================================|=====
** DBCC Status          |Errors Detected                   |*****
   DBCC Work units      |                                  |
   Dispatched           |75                                |
   DBCC Work units      |                                  |
   Completed            |75                                |
========================|==================================|=====
Index Summary           |                                  |
========================|==================================|=====
** Inconsistent Index   |                                  |
   Count                |1                                 |*****
   Verified Index       |                                  |
   Count                |85                                |
========================|==================================|=====
Index Statistics        |                                  |
========================|==================================|=====
** Inconsistent Index   |contact.DBA.idx01_HNG             |*****
   ...
   Verified Index       |fin_data.DBA.ASIQ_IDX_T209_C3_HG  |
   Verified Index       |fin_data.DBA.ASIQ_IDX_T209_C4_FP  |
   ...
   Verified Index       |employee.DBA.ASIQ_IDX_T212_C19_FP |
   Verified Index       |employee.DBA.ASIQ_IDX_T212_C20_FP |
```

```
   Verified Index       |iq_dummy.DBA.ASIQ_IDX_T213_C1_FP  |
** Extra Index RIDs     |5                                 |*****
   FP Indexes Checked   |68                                |
   HNG Indexes Checked  |1                                 |
   HG Indexes Checked   |17                                |
                        |                                  |
```

The inconsistent index detected by **sp_iqcheckdb** is `contact.DBA.idx01_HNG`.

The following DBCC output is generated when **sp_iqcheckdb** is run again to check just the inconsistent index. The command line executed for this example is `sp_iqcheckdb 'verify index DBA.contact.idx01_HNG'`.

```
         Stat                          Value                    Flags
=========================|==================================|=====
DBCC Verify Mode Report  |                                  |
=========================|==================================|=====
** DBCC Status           |Errors Detected                   |*****
   DBCC Work units       |                                  |
   Dispatched            |1                                 |
   DBCC Work units       |                                  |
   Completed             |1                                 |
                         |                                  |
=========================|==================================|=====
Index Summary            |                                  |
=========================|==================================|=====
** Inconsistent Index    |                                  |
   Count                 |1                                 |*****
   Verified Index        |                                  |
   Count                 |1                                 |
                         |                                  |
=========================|==================================|=====
Index Statistics         |                                  |
=========================|==================================|=====
** Inconsistent Index    |contact.DBA.idx01_HNG             |*****
   Verified Index        |contact.DBA.ASIQ_IDX_T206_C1_FP   |
** Extra Index RIDs      |5                                 |*****
   FP Indexes Checked    |1                                 |
   HNG Indexes Checked   |1                                 |
                         |                                  |
=========================|==================================|=====
```

DBCC Index Errors

The DBCC output contains messages related to problems with indexes.

Table 9. DBCC Index Errors

DBCC Message	Description/Action
Inconsistent Index Count	The number of indexes that DBCC found to have inconsistencies.
Inconsistent Index	The name of an index that DBCC found to be inconsistent.

DBCC Message	Description/Action
Extra Index RIDs Missing Index RIDs Duplicate Index RIDs	The total number of rows that are inconsistent for all inconsistent indexes.
Bitmap Verify Errors	The total number of inconsistent bitmaps in all database objects
FP Lookup Table Inconsistencies	An unrepairable error, where the 1-byte or 2-byte FP is internally inconsistent.
Non-Completed Index Count	The number of indexes that could not be verified, because an exception occurred while checking.
Non-Completed Index	The name of an index that was not verified because an exception occurred while checking. If the exception is a future version, out of memory, or out of buffers error, commit the DBCC connection and re-run DBCC.
VDO Incorrect First Available Fields VDO Incorrect Next Available Fields VDO Incorrect Used Count Fields VDO Incorrect In-use Bitvec VDO Incorrect In-use Bitmap VDO Incorrect Partial Bitmap VDO Incorrect Deleted Bitmaps	Unrepairable errors that can cause entire tables to be inaccessible. You must force drop the inconsistent table to resolve these errors.
HG Missing Groups HG Extra Groups HG Extra Keys HG Missing Keys B-Tree Invalid Item Count B-Tree Invalid Item Count G-Array Empty Page Errors G-Array Bad Group Type Errors G-Array Out of Order Group Errors	High Group index specific errors.

Index Error Repair

Use the **sp_iqrebuildindex** procedure to repair an index, then run **sp_iqcheckdb** in verify mode to check for index inconsistencies.

If an index is still inconsistent, drop and recreate the index, and then rebuild the index.

Note: The **sp_iqrebuildindex** procedure cannot repair FP indexes. SAP Sybase IQ has no functionality to repair FP indexes.

Analysis of Allocation Problems

Use **sp_iqcheckdb** to analyze allocation problems.

The database maintains an allocation map, also known as a free list, which tracks the blocks that are in use by database objects.

DBCC detects three types of allocation problems:

- Leaked blocks—A leaked block is a block that is allocated according to the database allocation map, but is found not to be part of any database objects. DBCC can recover leaked blocks.
- Unallocated blocks—An unallocated block is a block that is not allocated according to the database allocation map, but is found to be in use by a database object. DBCC can recover unallocated blocks.
- Multiply-owned blocks—A multiply-owned block is a block that is in use by more than one database object. At least one of the structures involved contains inconsistent data. DBCC cannot repair this type of allocation problem. If you encounter this type of error, run DBCC again, specifying a list of indexes, until you identify the indexes that share the block. These indexes must then all be dropped to eliminate the multiply-owned block.

Sample of Leaked Space Output

This is an example of the output you see when you run **sp_iqcheckdb** and there is leaked space. Lines with asterisks (*****) contain information about allocation problems. In this example, DBCC reports 16 leaked blocks.

The command line executed for this example is `sp_iqcheckdb 'allocation database'`.

```
          Stat                            Value                 Flags
==============================|================================|=====
DBCC Allocation Mode Report   |                                |
==============================|================================|=====
** DBCC Status                |Errors Detected                 |*****
   DBCC Work units Dispatched |164                             |
   DBCC Work units Completed  |164                             |
                              |                                |
==============================|================================|=====
Allocation Summary            |                                |
==============================|================================|=====
```

```
   Blocks Total                |8192                             |
   Blocks in Current Version   |4785                             |
   Blocks in All Versions      |4785                             |
   Blocks in Use               |4801                             |
   % Blocks in Use             |58                               |
** Blocks Leaked               |16                               |*****
                               |                                 |
===============================|=================================|=====
Allocation Statistics          |                                 |
===============================|=================================|=====
   ...
** 1st Unowned PBN             |1994                             |*****
   ...
===============================|=================================|=====
```

If one or more dbspaces are offline, use the following syntax to show allocation problems for a particular dbspace:

```
sp_iqcheckdb 'allocation dbspace dbspace-name'
```

DBCC Allocation Errors

Allocation problems are reported in the output generated by DBCC with **sp_iqcheckdb** run in a allocation mode or verification mode. If the Allocation Summary section has values flagged with asterisks, such as "** Blocks Leaked" or "** Blocks with Multiple Owners," then there are allocation problems.

Messages in the DBCC output related to allocation problems are listed in the following table.

Table 10. DBCC Allocation Errors

DBCC Message	Description/Action
Block Count Mismatch	This count always accompanies other allocation errors.
Blocks Leaked 1st Unowned PBN	Blocks that were found not to be in use by any database object. Use **sp_iqcheckdb dropleaks** mode to repair.
Blocks with Multiple Owners 1st Multiple Owner PBN	Blocks in use by more than one database object. Drop the object that is reported as inconsistent.
Unallocated Blocks in Use 1st Unallocated PBN	Blocks in use by a database object, but not marked as in use. Use **sp_iqcheckdb dropleaks** mode to repair.

If the Allocation Summary lines indicate no problem, but the Index Summary section reports a value for "Inconsistent Index Count," then this indicates one or more inconsistent indexes.

Repairing Allocation Problems using DBCC

Use **sp_iqcheckdb dropleaks** to repair database allocation problems.

Note: This procedure uses the **-gd** and **-gm** switches to restrict database access. For a more restrictive method, start in forced recovery mode.

1. Start the server.

   For example:

   ```
   start_iq -n my_db_server -x 'tcpip{port=7934}'
   -gd dba -gm 1 /work/database/my_db.db
   ```

   Note: You must start the database with the ".db" extension, not ".DB".

   Use two server startup switches to restrict access:

   - Use **-gd DBA** so that only users with the SERVER OPERATOR system privilege can start and stop databases. (Note that the client must already have a connection to the server to start or stop the database, so this switch does not prevent connections.)
   - Use **-gm 1** to allow a single connection plus one DBA connection above the limit so that a DBA can connect and drop others in an emergency.

2. Run the stored procedure **sp_iqcheckdb** in dropleaks mode:

   ```
   sp_iqcheckdb 'dropleaks database'
   ```

   If one or more dbspaces are offline, you can repair allocation problems for a dbspace alone by running:

   ```
   sp_iqcheckdb 'dropleaks dbspace dbspace-name'
   ```

   If the allocation repair is successful, **sp_iqcheckdb** displays the message "Freelist Updated." If errors are detected, **sp_iqcheckdb** returns the messages "Freelist Not Updated" and "Errors Detected."

3. Stop the server after **sp_iqcheckdb** finishes. To stop the server, use **stop_iq** on UNIX or the shutdown button in the console window on Windows.

After allocation problems are repaired, allocation statistics appear in the DBCC output with no errors.

DBCC displays an Allocation Summary section at the top of the report, which lists information about allocation usage. The Allocation Statistics section provides more details about the blocks. The DBCC output does not contain repair messages for the leaked blocks that have been recovered.

For example:

```
sp_iqcheckdb 'dropleaks dbspace mydbspace';
checkpoint;
```

The **sp_iqcheckdb** output indicates no errors, so the **checkpoint** is executed.

DBCC reports statistics that do not show in this abbreviated output.

```
          Stat                              Value                       Flags
================================|===================================|
=====
DBCC Allocation Mode Report     |                                   |
================================|===================================|
=====
   DBCC Status                  |Freelist Updated                    |
   DBCC Status                  |No Errors Detected                  |
   DBCC Work units Dispatched   |75                                  |
   DBCC Work units Completed    |75                                  |
================================|===================================|
=====
Allocation Summary              |                                   |
================================|===================================|
=====
   Blocks Total                 |8192                                |
   Blocks in Current Version    |4594                                |
   Blocks in All Versions       |4594                                |
   Blocks in Use                |4610                                |
   % Blocks in Use              |56                                  |
================================|===================================|
=====
Allocation Statistics           |                                   |
================================|===================================|
=====
   DB Extent Count              |1                                   |
   Marked Logical Blocks        |8176                                |
   Marked Physical Blocks       |4594                                |
   Marked Pages                 |511                                 |
   Blocks in Freelist           |126177                              |
   Imaginary Blocks             |121567                              |
   Highest PBN in Use           |5425                                |
   Total Free Blocks            |3582                                |
   Usable Free Blocks           |3507                                |
   % Free Space Fragmented      |2                                   |
   Max Blocks Per Page          |16                                  |
   1  Block Page Count          |103                                 |
   3  Block Page Count          |153                                 |
   ...
   16 Block Hole Count          |213                                 |
================================|===================================|
=====
```

Note: When performing forced recovery or leaked blocks recovery, you must start the database with the ".db" extension, not ".DB". For example:

```
 start_iq -n my_db_server -x 'tcpip{port=7934}'
-gd dba -iqfreq my_db /work/database/my_db.db
```

Forced Recovery Mode

Forced database recovery differs from normal database recovery.

- **Forced recovery marks all storage within the database as in use –** In order to recover a potentially inconsistent allocation map, all storage within the database is marked as in use. Use the **sp_iqcheckdb** in **dropleaks** mode to reset the allocation map to the correct state.
- **Incremental backups are disabled –** After the database is opened in forced recovery mode, incremental backups are disabled. The next backup must be a full backup. Doing a full backup reenables incremental backups.
- **The forced recovery parameter applies to all opens of the database while the server is up –** Therefore, after the database is opened, the DBA needs to bring the server back down, and then restart the server without the forced recovery flag, to be sure that subsequent opens run in regular mode. Repeated opens of the database with forced recovery on do not harm the database, but could be confusing to the DBA. Each time you open the database in forced recovery mode, all the storage within the database is marked as in use.

Before Forced Recovery

Restricting database access provides greater control over inadvertent database opens during forced recovery.

Use two server startup switches to restrict access:

- Use **-gd DBA** so that only users with the SERVER OPERATOR system privilege can start and stop databases on a running server. (Note that the client must already have a connection to the server to start or stop the database, so this switch does not prevent connections.)
- Use **-gm 1** to allow a single connection plus one DBA connection above the limit so that a DBA can connect and drop others in an emergency.

An alternate way to restrict connections is to specify:

```
sa_server_option('disable_connections', 'ON')
```

just after you start the connection where you are performing forced recovery and

```
sa_server_option('disable_connections', 'OFF')
```

on the same connection after recovery. The disadvantage is that this method precludes emergency access from another DBA connection.

Starting a Server in Forced Recovery Mode

Forced recovery allows the server to start if the allocation map is inconsistent.

If you cannot start a server or database in a multiplex, forced recovery may be needed. Use forced recovery only when normal database recovery fails to restore the database to a running

state, and only if you see `s_buf` or free list errors during recovery. Never use forced recovery in response to SQL Anywhere errors, such as SA transaction log replay failure.

If you have followed documented recovery procedures and SAP Sybase Technical Support recommends forced recovery, follow these steps:

1. Shut down all secondary nodes using **stop_iq**.
2. Start the server with the **-iqfrec** and **-iqmpx_sn 1** flags:

```
start_iq -n my_server -x 'tcpip(port=7934}'
-gd dba -gm 1 -iqmpx_sn 1 -iqfrec
my_db /database/my_db.db
```

3. Connect to the server and run:

```
sp_iqcheckdb 'dropleaks database'
checkpoint
```

4. Correct errors and rerun **sp_iqcheckdb**. Repeat until no errors result.
5. Shut down and restart the server normally (without the flags in Step 2).

If you cannot start your server in forced recovery mode, contact SAP Sybase Technical Support.

Using Forced Recovery without a Follow On sp_iqcheckdb

Running forced recovery starts the database in a valid, but fully allocated mode. In other words, you should be able to do all operations, but no permanent main dbspace is left. Before you do anything else, you must either recover the lost dbspace by running **sp_iqcheckdb** in **dropleaks** mode, or add a new dbspace. Note that queries should also run successfully, since they do not need additional permanent dbspace; however, you cannot load, insert, or delete data.

Warning! Running queries without verifying the database will not cause any inconsistency in your data. However, if there is a problem in the data that caused the server to fail, the server could fail again or produce incorrect results.

Recovering Leaked Space

Use the **sp_iqcheckdb** stored procedure in **dropleaks** mode to recover leaked storage space within the specified database.

An allocation map is used by the server to determine if a page is in use or not in use within IQ. Either through system failure or as a result of opening a database with forced recovery, the allocation map of the database may not reflect the true allocation of its usage. When this occurs, we say that the database has “leaked” storage or “leaked blocks.” In general, you need not be concerned about small numbers of leaked blocks. If you have many megabytes of leaked blocks, you probably want to recover that space.

When leaked storage is being recovered, other transactions that alter the allocation map are shut out. Such operations include checkpoints and commands that modify the database.

You can recover leaked storage and force recovery either at the same time or separately. To recover leaked space within a database without doing a forced recovery, repair allocation problems using DBCC. To recover leaked space within a database after doing a forced recovery, recover leaked space using this procedure.

If repairing allocation problems using DBCC fails to recover leaked storage, then use this procedure.

Note: This procedure uses the **-gd** and **-gm** switches to restrict database access. For a more restrictive method, start the server in forced recovery mode.

1. Start the server with the `-iqfrec` option in the **start_iq** command.

   ```
   start_iq -n my_db_server -x 'tcpip{port=7934}'
   -gd dba -gm 1
   -iqfrec my_db /work/database/my_db.db
   ```

   You specify the database name twice in a row, once to specify it as the database you are starting, and once to specify it as the database undergoing forced recovery. The **-iqfrec** option requires the database name.
2. Connect to the database you are recovering.
3. Run the stored procedure **sp_iqcheckdb** in dropleaks mode.

   ```
   sp_iqcheckdb 'dropleaks database'
   ```

   If there are no errors and **sp_iqcheckdb** displays the message `Freelist Updated`, you have recovered leaked space and forced recovery. Continue to the next step.

   If inconsistency is found, drop inconsistent indexes, tables, or columns. Then run **sp_iqcheckdb** again to recover leaked space.
4. Issue a checkpoint.
5. Stop the server using your usual method.
6. Restart the server using your usual method, and proceed with normal processing.

Recovering Multiplex Databases

Multiplex databases have special recovery requirements.

Before recovering a multiplex database, see *Administration: Multiplex*.

Problems Reported by DBCC

Messages are reported for problems that DBCC cannot repair.

Table 11. Messages for Problems DBCC Cannot Repair

DBCC message	Description/action
FP Lookup Table Inconsistencies	An unrepairable error, where the 1-byte or 2-byte FP is internally inconsistent.
VDO Incorrect First Available Fields VDO Incorrect Next Available Fields VDO Incorrect Used Count Fields VDO Incorrect In-use Bitvec VDO Incorrect In-use Bitmap VDO Incorrect Partial Bitmap VDO Incorrect Deleted Bitmaps	Unrepairable errors that can cause entire tables to be inaccessible. You must force drop the inconsistent table to resolve these errors.
Blocks with Multiple Owners 1st Multiple Owner PBN	Blocks in use by more than one database object. Drop the object that is reported as inconsistent.
DBCC Meta-data Errors Blockmap Invalid Chunksize Error Count Blockmap Compression Bit Error Count Blockmap Invalid Block Number Error Count	An internal page mapping structure is inconsistent and the object needs to be dropped.
DBCC Inconsistent Disk Block Headers DBCC Decompress Errors	The storage for the object is inconsistent and the object needs to be dropped.

Index Problems that DBCC Cannot Repair

Use these suggestions to repair inconsistent indexes.

If DBCC detects a problem with an index, the name of the index is reported with the type of problem. Use **sp_iqrebuildindex** to repair a non-FP index. FP indexes cannot be repaired. Analyze index errors for indexes reported as “Inconsistent Index,” when **sp_iqcheckdb** is run in default or check mode.

Depending on the type of problem, use **DROP INDEX**, **ALTER TABLE DROP COLUMN**, **DROP TABLE**, or the **FORCE_DROP** option to resolve the problem.

Call SAP Sybase Technical Support for help in determining the best course of action to fix an inconsistent index or table.

Dropping Inconsistent Indexes, Tables, or Columns

Use these suggestions to resolve issues with unrepairable indexes, columns, or tables.

If **sp_iqcheckdb** reports unrepairable indexes, columns, or tables, then these objects must be dropped using the **DROP INDEX**, **ALTER TABLE DROP COLUMN**, or **DROP TABLE** statements respectively.

Note: You should not attempt to force drop objects unless Sybase Technical Support has instructed you to do so.

If you cannot drop an inconsistent object, set the temporary `FORCE_DROP` option. `FORCE_DROP` causes the IQ server to silently leak the on-disk storage of the dropped object, rather than try to reclaim it. You can recover the leaked space later using DBCC. This is desirable for an inconsistent object, because the only information about the storage of an object is within the object itself, and this information is suspect for an inconsistent object.

The `FORCE_DROP` database option is not allowed on a secondary node. If a force drop is attempted on a secondary node, an error is returned. `FORCE_DROP` is a temporary option, so that the value of the option does not get propagated to secondary nodes at synchronization.

Note: When force dropping objects, you must ensure that only the DBA is connected to the database. Restart the server immediately after a force drop.

The following procedure uses the **-gd** and **-gm** switches to restrict database access. The **-gd** switch only limits users who can start or stop databases on a running server. For a more restrictive method, start the server in forced recovery mode.

1. Restart the server.

   ```
   start_iq -n bad_db_server -x 'tcpip{port=7934}'
   -gm 1 -gd dba bad_db.db
   ```

   You must not allow other users to connect when force dropping objects.

   Use two server startup switches to restrict access:

   - Use **-gd DBA** so that only users with the SERVER OPERATOR system privilege can start and stop databases. (Note that the client must already have a connection to the server to start or stop the database, so this switch does not prevent connections.)
   - Use **-gm 1** to allow a single connection plus one DBA connection above the limit so that a DBA can connect and drop others in an emergency.

   For more information about restricting connections, see *Installation and Configuration Guide*.

2. Set the temporary option `FORCE_DROP` to ON.

   ```
   set temporary option FORCE_DROP = 'ON'
   ```

3. Drop all inconsistent objects.

Use the commands **DROP INDEX**, **ALTER TABLE DROP COLUMN**, or **DROP TABLE** as needed. Do not enter any other DDL or DML commands until after restarting the server.

4. Restart the server.

   To recover the leaked space and update the allocation map to the correct state, start the server.

   ```
   start_iq -n bad_db_server -x 'tcpip{port=7934}'
   -gm 1 -gd dba bad_db.db
   ```

5. Run sp_iqcheckdb.

   ```
   sp_iqcheckdb 'dropleaks database';
   ```

   This step resets the database allocation map to the calculated allocation map.

DBCC Error Messages

These are the most important messages in the DBCC output.

Table 12. DBCC Error Messages

DBCC Message	Description/Aaction
Inconsistent Index Count	The number of indexes that DBCC found to have inconsistencies.
Inconsistent Index	The name of an index that DBCC found to be inconsistent.
Extra Index RIDs Missing Index RIDs Duplicate Index RIDs	The total number of rows that are inconsistent for all inconsistent indexes.
Bitmap Verify Errors	The total number of inconsistent bitmaps in all database objects.
FP Lookup Table Inconsistencies	An unrepairable error, where the 1-byte or 2-byte FP is internally inconsistent.
Non-Completed Index Count	The number of indexes that could not be verified, because an exception occurred while checking.
Non-Completed Index	The name of an index that was not verified because an exception occurred while checking. If the exception is a future version, out of memory, or out of buffers error, commit the DBCC connection and re-run DBCC.
HG Missing Groups HG Extra Groups HG Extra Keys HG Missing Keys B-Tree Invalid Item Count B-Tree Invalid Item Count G-Array Empty Page Errors G-Array Bad Group Type Errors G-Array Out of Order Group Errors	High Group index specific errors.

DBCC Message	Description/Aaction
VDO Incorrect First Available Fields VDO Incorrect Next Available Fields VDO Incorrect Used Count Fields VDO Incorrect In-use Bitvec VDO Incorrect In-use Bitmap VDO Incorrect Partial Bitmap VDO Incorrect Deleted Bitmaps	Unrepairable errors that can cause entire tables to be inaccessible. You must force drop the inconsistent table to resolve these errors.
Block Count Mismatch	This count accompanies other allocation errors.
Blocks Leaked 1st Unowned PBN	Blocks that were found not to be in use by any database object. Use dropleaks mode to repair.
Blocks with Multiple Owners 1st Multiple Owner PBN	Blocks in use by more than one database object. Drop the object that is reported as inconsistent.
Unallocated Blocks in Use 1st Unallocated PBN	Blocks in use by a database object, but not marked as in use. Use dropleaks mode to repair.
Freelist Updated	Indicates successful allocation repair.
Freelist Not Updated	Indicates errors detected during allocation repair and the allocation repair was not successful.
Invalid Blockmap Unique ID Generator Blockmap Unique ID Generator Updated Invalid Transaction ID Counter Transaction ID Generator Updated	Errors and repair messages specific to the DBCC **resetclocks** option.
DBCC Future Version Errors	DBCC could not open the table, because DDL was performed on it. Commit the DBCC connection and re-run DBCC.
DBCC Locked Table Access Conflict	DBCC tried to open a table that another connection has locked. To ensure complete DBCC processing, make sure that no other users have locked tables in the database.
DBCC Out of Buffers Errors	The size of the IQ main cache is too small. Either increase the main cache size or run DBCC on individual objects.
DBCC Out of Memory Errors	There is insufficient system memory to complete the DBCC operation.
DBCC Meta-data Errors Blockmap Invalid Chunksize Error Count Blockmap Compression Bit Error Count Blockmap Invalid Block Number Error Count	An internal page mapping structure is inconsistent and the object needs to be dropped.
DBCC Page Read Errors	An I/O error occurred while trying to read an object. Perform hardware diagnostics.

DBCC Message	Description/Aaction
DBCC Inconsistent Disk Block Headers DBCC Decompress Errors	The storage for the object is inconsistent and the object needs to be dropped.
DBCC Unknown Exceptions	An exception of a type unknown to DBCC occurred. Check the IQ message file for details.
Unowned LVC cells Duplicate LVC cell rows Unallocated LVC cell rows	Messages indicate inconsistencies with a `VARCHAR` or `CLOB` column. Unowned LVC cells represent a small amount of unusable disk space and can safely be ignored. Duplicate and Unallocated LVC cells are serious errors that can only be resolved by dropping the damaged columns. To drop a damaged column, create a new column from a copy of the old column, then drop the original column and alter rename the new column to the old column. LVC is a `VARCHAR` column with a width greater than 255. `CLOB` also uses LVC.
Hash Pid: '%pid' is corrupt, count mismatch. Missing RIDs in RID mgr Hash Partition corruption, RID range mismatch	An error in the hash or hash-range load or insertion assigned row IDs incorrectly from different hash partitions. Unload and reload the table.

Backup Reference

Certain SQL statements have special syntax to support backup and restore operations.

BACKUP DATABASE Statement

Backs up an SAP Sybase IQ database on one or more archive devices.

Quick Links:

Go to Parameters on page 67

Go to Examples on page 70

Go to Usage on page 70

Go to Standards on page 73

Go to Permissions on page 73

Syntax

```
BACKUP DATABASE
    [ backup-option … ]
    TO archive_device [ archive-option . . . ]
    … [ WITH COMMENT string ]

backup-option – (back to Syntax)
    { READWRITE FILES ONLY |
    READONLY dbspace-or-file [, … ] }
    CRC { ON | OFF }
    ATTENDED { ON | OFF }
    BLOCK FACTOR integer
    { FULL | INCREMENTAL | INCREMENTAL SINCE FULL }
    VIRTUAL { DECOUPLED |
    ENCAPSULATED 'shell_command' }
    WITH COMMENT comment

dbspace-or-file – (back to backup-option)
    { DBSPACES identifier-list | FILES identifier-list }

identifier-list – (back to dbspace-or-file)
    identifier [, … ]

archive-option – (back to Syntax)
    SIZE integer STACKER integer
```

Parameters

(back to top) on page 67

- **TO –** specify the name of the archive_device to be used for backup, delimited with single quotation marks. The archive_device is a file name or tape drive device name for the archive file. If you use multiple archive devices, specify them using separate TO clauses. (A comma-separated list is not allowed.) Archive devices must be distinct. The number of TO clauses determines the amount of parallelism SAP Sybase IQ attempts with regard to output devices.
- **WITH COMMENT –** specify an optional comment recorded in the archive file and in the backup history file. Maximum length is 32KB. If you do not specify a value, a NULL string is stored.
- **READWRITE FILES ONLY –** restricts FULL, INCREMENTAL, and INCREMENTAL SINCE FULL backups to only the set of read-write files in the database. The read-write dbspaces/files must be SAP Sybase IQ dbspaces.

  If READWRITE FILES ONLY clause is used with an INCREMENTAL or INCREMENTAL SINCE FULL backup, the backup will not back up data on read-only dbspaces or dbfiles that has changed since the depends-on backup. If READWRITE FILES ONLY is not specified for an INCREMENTAL or INCREMENTAL SINCE FULL backup, the backup backs up all database pages that have changed since the depends-on backup, both on read-write and read-only dbspaces.
- **CRC –** activates 32-bit cyclical redundancy checking on a per block basis (in addition to whatever error detection is available in the hardware). When you specify this clause, the numbers computed on backup are verified during any subsequent restore operation, affecting performance of both commands. The default is ON.
- **ATTENDED –** applies only when backing up to a tape device. If ATTENDED ON clause (the default) is used, a message is sent to the application that issued the **BACKUP DATABASE** statement if the tape drive requires intervention. This might happen, for example, when a new tape is required. If you specify OFF, **BACKUP DATABASE** does not prompt for new tapes. If additional tapes are needed and OFF has been specified, SAP Sybase IQ gives an error and aborts the **BACKUP DATABASE** command. However, a short delay is included to account for the time an automatic stacker drive requires to switch tapes.
- **BLOCK FACTOR *integer* –** specify the number of blocks to write at one time. The value must be greater than 0, or SAP Sybase IQ generates an error message. Its default is 25 for UNIX systems and 15 for Windows systems (to accommodate the smaller fixed tape block sizes). This clause effectively controls the amount of memory used for buffers. The actual amount of memory is this value times the block size times the number of threads used to extract data from the database. Set BLOCK FACTOR to at least 25.
- **FULL | INCREMENTAL | INCREMENTAL SINCE FULL –**
  - **FULL –** specify a full backup; all blocks in use in the database are saved to the archive devices. This is the default action.

  - **INCREMENTAL –** specify an incremental backup; all blocks changed since the last backup of any kind are saved to the archive devices. The keyword **INCREMENTAL** is not allowed with **READONLY FILES**.
  - **INCREMENTAL SINCE FULL –** specify an incremental backup; all blocks changed since the last full backup are saved to the archive devices.
- **VIRTUAL DECOUPLED –** specify a decoupled virtual backup. For the backup to be complete, you must copy the SAP Sybase IQ dbspaces after the decoupled virtual backup finishes, and then perform a nonvirtual incremental backup.
- **VIRTUAL ENCAPSULATED –** specify an encapsulated virtual backup. The 'shell-command' argument can be a string or variable containing a string that is executed as part of the encapsulated virtual backup. The shell commands execute a system-level backup of the IQ store as part of the backup operation. For security reasons, it is recommended that an absolute path be specified in the 'shell-command,' and file protections on that directory be in place to prevent execution of an unintended program.
- **SIZE –** Specify maximum tape or file capacity per output device (some platforms do not reliably detect end-of-tape markers). No volume used on the corresponding device should be shorter than this value. This value applies to both tape and disk files but not third-party devices. Units are kilobytes (KB), although in general, less than 1GB is inappropriate. For example, for a 3.5GB tape, specify 3500000. Defaults are by platform and medium. The final size of the backup file will not be exact, because backup writes in units of large blocks of data.

Table 13. BACKUP DATABASE default sizes

Platform	Default SIZE for Tape	Default SIZE for Disk
UNIX	none	2GB
Windows	1.5GB SIZE must be a multiple of 64. Other values are rounded down to a multiple of 64.	1.5GB

The SIZE parameter is per output device. SIZE does not limit the number of bytes per device; SIZE limits the file size. Each output device can have a different SIZE parameter. During backup, when the amount of information written to a given device reaches the value specified by the SIZE parameter, **BACKUP DATABASE** does one of the following:

  - If the device is a file system device, **BACKUP DATABASE** closes the current file and creates another file of the same name, with the next ascending number appended to the file name, for example, bkup1.dat1.1, bkup1.dat1.2, bkup1.dat1.3.
  - If the device is a tape unit, **BACKUP DATABASE** closes the current tape and you need to mount another tape.
- **STACKER –** specify that the device is automatically loaded, and specifies the number of tapes with which it is loaded. This value is not the tape position in the stacker, which could be zero. When ATTENDED is OFF and STACKER is ON, SAP Sybase IQwaits for a predetermined amount of time to allow the next tape to be autoloaded. The number of tapes

supplied along with the SIZE clause are used to determine whether there is enough space to store the backed-up data. Do not use this clause with third-party media management devices.

Examples

(back to top) on page 67

- **Example 1 –** this UNIX example backs up the `iqdemo` database onto tape devices `/dev/rmt/0` and `/dev/rmt/2` on a Sun Solaris platform. On Solaris, the letter `n` after the device name specifies the "no rewind on close" feature. Always specify this feature with **BACKUP DATABASE**, using the naming convention appropriate for your UNIX platform (Windows does not support this feature). This example backs up all changes to the database since the last full backup:

```
BACKUP DATABASE
INCREMENTAL SINCE FULL
TO '/dev/rmt/0n' SIZE 10000000
TO '/dev/rmt/2n' SIZE 15000000
```

Note: Size units are kilobytes (KB), although in most cases, size of less than 1GB are inappropriate. In this example, the specified sizes are 10GB and 15GB.

- **Example 2 –** these **BACKUP DATABASE** commands specify read-only files and dbspaces:

```
BACKUP DATABASE READONLY DBSPACES dsp1
TO '/dev/rmt/0'
```

```
BACKUP DATABASE READONLY FILES dsp1_f1, dsp1_f2
TO 'bkp.f1f2'
```

```
BACKUP DATABASE READONLY DBSPACES dsp2, dsp3
READONLY FILES dsp4_f1, dsp5_f2
TO 'bkp.RO'
```

Usage

(back to top) on page 67

The SAP Sybase IQ database might be open for use by many readers and writers when you execute a **BACKUP DATABASE** command. It acts as a read-only user and relies on the Table Level Versioning feature of SAP Sybase IQ to achieve a consistent set of data.

BACKUP DATABASE implicitly issues a **CHECKPOINT** prior to commencing, and then it backs up the catalog tables that describe the database (and any other tables you have added to the catalog store). During this first phase, SAP Sybase IQ does not allow any metadata changes to the database (such as adding or dropping columns and tables). Correspondingly, a later **RESTORE DATABASE** of the backup restores only up to that initial **CHECKPOINT**.

The **BACKUP DATABASE** command lets you specify full or incremental backups. You can choose two kinds of incremental backups. INCREMENTAL backsup only those blocks that have changed and committed since the last backup of any type (incremental or full).

INCREMENTAL SINCE FULL backs up all of the blocks that have changed since the last full backup. The first type of incremental backup can be smaller and faster to do for **BACKUP DATABASE** commands, but slower and more complicated for **RESTORE DATABASE** commands. The opposite is true for the other type of incremental backup. The reason is that the first type generally results in N sets of incremental backup archives for each full backup archive. If a restore is required, a user with the SERVER OPERATOR system privilege must restore the full backup archive first, and then each incremental archive in the proper order. (SAP Sybase IQ keeps track of which ones are needed.) The second type requires the user with the SERVER OPERATOR system privilege to restore only the full backup archive and the last incremental archive.

Incremental virtual backup is supported using the VIRTUAL DECOUPLED and VIRTUAL ENCAPSULATED parameters of the **BACKUP DATABASE** statement.

Although you can perform an OS-level copy of tablespaces to make a virtual backup of one or more read-only dbspaces, use the virtual backup statement, because it records the backup in the SAP Sybase IQ system tables.

BACKUP DATABASE and **RESTORE DATABASE** write your SAP Sybase IQ data in parallel to or from all of the archive devices you specify. The catalog store is written serially to the first device. Faster backups and restores result from greater parallelism.

SAP Sybase IQ supports a maximum of 36 hardware devices for backup. For faster backups, specifying one or two devices per core will help to avoid hardware and IO contention. Set the SIZE parameter on the **BACKUP DATABASE** command to avoid creating multiple files per backup device and consider the value used in the BLOCK FACTOR clause on the **BACKUP DATABASE** command.

BACKUP DATABASE overwrites existing archive files unless you move the old files or use a different *archive_device* name or path.

The backup API DLL implementation lets you specify arguments to pass to the DLL when opening an archive device. For third-party implementations, the archive_device string has this format:

```
'DLLidentifier::vendor_specific_information'
```

A specific example:

```
'spsc::workorder=12;volname=ASD002'
```

The *archive_device* string length can be up to 1023 bytes. The *DLLidentifier* portion must be 1 to 30 bytes in length and can contain only alphanumeric and underscore characters. The *vendor_specific_information* portion of the string is passed to the third-party implementation without checking its contents. Do not specify the SIZE or STACKER clauses of the **BACKUP DATABASE** command when using third-party implementations, as that information should be encoded in the *vendor_specific_information* portion of the string.

Note: Only certain third-party products are certified with SAP Sybase IQ using this syntax. See the *Release Bulletin* for additional usage instructions or restrictions. Before using any

third-party product to back up your SAP Sybase IQ database in this way, make sure it is certified. See the *Release Bulletin*, or see the SAP Sybase Certification Reports for the SAP Sybase IQ product in *Technical Documents at http://www.sybase.com/support/techdocs/*.

For the SAP Sybase IQ implementation of the backup API, you need to specify only the tape device name or file name. For disk devices, you should also specify the SIZE value, or SAP Sybase IQ assumes that each created disk file is no larger than 2GB on UNIX, or 1.5GB on Windows.

An example of an archive device for the SAP Sybase API DLL that specifies a tape device for certain UNIX systems is:

```
'/dev/rmt/0'
```

It is your responsibility to mount additional tapes if needed, or to ensure that the disk has enough space to accommodate the backup.

When multiple devices are specified, **BACKUP DATABASE** distributes the information across all devices. Other issues for **BACKUP DATABASE** include:

- **BACKUP DATABASE** does not support raw devices as archival devices.
- Windows systems support only fixed-length I/O operations to tape devices (for more information about this limitation, see your *Installation and Configuration Guide*). Although Windows supports tape partitioning, SAP Sybase IQ does not use it, so do not use another application to format tapes for **BACKUP DATABASE**. Windows has a simpler naming strategy for its tape devices, where the first tape device is `\\.\`*tape0*, the second is `\\.\`*tape1*, and so on.

  Warning! For backup (and for most other situations) SAP Sybase IQ treats the leading backslash in a string as an escape character, when the backslash precedes an n, an x, or another backslash. For this reason, when you specify backup tape devices, you must double each backslash required by the Windows naming convention. For example, indicate the first Windows tape device you are backing up to as '`\\\\.\\tape0`', the second as '`\\\\.\\tape1`', and so on. If you omit the extra backslashes, or otherwise misspell a tape device name, and write a name that is not a valid tape device on your system, SAP Sybase IQ interprets this name as a disk file name.
- SAP Sybase IQ does not rewind tapes before using them. You must ensure the tapes used for backup and restore are at the correct starting point before putting them in the tape device. SAP Sybase IQ does rewind tapes after using them on rewinding devices.
- During backup and restore operations, if SAP Sybase IQ cannot open the archive device (for example, when it needs the media loaded) and the ATTENDED clause is ON, it waits for ten seconds and tries again. It continues these attempts indefinitely until either it is successful or the operation is terminated with a Ctrl+C.
- If you enter Ctrl+C, **BACKUP DATABASE** fails and returns the database to the state it was in before the backup started.
- If disk striping is used, such as on a RAID device, the striped disks are treated as a single device.

Side effects:

- Automatic commit

Standards

(back to top) on page 67

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- SAP Sybase Database product—Not supported by SAP Sybase Adaptive Server® Enterprise.

Permissions

(back to top) on page 67

Requires one of:

- BACK UP DATABASE system privilege.
- You own the database.

RESTORE DATABASE Statement

Restores an SAP Sybase IQ database backup from one or more archive devices.

Quick Links:

Go to Parameters on page 74

Go to Examples on page 76

Go to Usage on page 77

Go to Standards on page 79

Go to Permissions on page 79

Syntax

Syntax 1

```
RESTORE DATABASE 'db_file'
    'archive_device' [ FROM 'archive_device' ]…
   … [ CATALOG ONLY ]
   … [ KEY key_spec ]
   … [ [ RENAME logical-dbfile-name TO 'new-dbspace-path'] ...
       | VERIFY [ COMPATIBLE ] ]
```

Syntax 2

```
RESTORE DATABASE 'database-name'
   [ restore-option ...]
   FROM 'archive_device' ...
```

```
restore-option
    READONLY dbspace-or-file [, … ]
    KEY key_spec
    RENAME file-name TO new-file-path ...
```

Parameters

(back to top) on page 73

- **db_file –** relative or absolute path of the database to be restored. Can be the original location, or a new location for the catalog store file.
- **key_spec –** quoted string including mixed cases, numbers, letters, and special characters. It might be necessary to protect the key from interpretation or alteration by the command shell.
- **FROM –** specifies the name of the *archive_device* from which you are restoring, delimited with single quotation marks. If you are using multiple archive devices, specify them using separate FROM clauses. A comma-separated list is not allowed. Archive devices must be distinct. The number of FROM clauses determines the amount of parallelism SAP Sybase IQ attempts with regard to input devices.

  The backup/restore API DLL implementation lets you specify arguments to pass to the DLL when opening an archive device. For third-party implementations, the *archive_device* string has this format:

  ```
  'DLLidentifier::vendor_specific_information'
  ```

  A specific example is:

  ```
  'spsc::workorder=12;volname=ASD002'
  ```

  The *archive_device* string length can be up to 1023 bytes. The *DLLidentifier* portion must be 1 to 30 bytes in length and can contain only alphanumeric and underscore characters. The *vendor_specific_information* portion of the string is passed to the third-party implementation without checking its contents.

  Note: Only certain third-party products are certified with SAP Sybase IQ using this syntax. See the *Release Bulletin* for additional usage instructions or restrictions. Before using any third-party product to back up your SAP Sybase IQ database, make sure it is certified. See the *Release Bulletin*, or see the SAP Sybase IQ Certification Reports for the SAP Sybase IQ product in *Technical Documents*.

  For the SAP Sybase IQ implementation of the backup/restore API, you need not specify information other than the tape device name or file name. However, if you use disk devices, you must specify the same number of archive devices on the restore as given on the backup; otherwise, you may have a different number of restoration devices than the number used to perform the backup. A specific example of an archive device for the SAP Sybase IQ API DLL that specifies a nonrewinding tape device for a UNIX system is:

  ```
  '/dev/rmt/0n'
  ```

- **CATALOG ONLY –** restores only the backup header record from the archive media.
- **RENAME –** restore one or more SAP Sybase IQ database files to a new location. Specify each *dbspace-name* you are moving as it appears in the `SYSFILE` table. Specify *new-dbspace-path* as the new raw partition, or the new full or relative path name, for that dbspace.

  If relative paths were used to create the database files, the files are restored by default relative to the catalog store file (the `SYSTEM` dbspace), and a rename clause is not required. If absolute paths were used to create the database files and a rename clause is not specified for a file, it is restored to its original location.

  Relative path names in the RENAME clause work as they do when you create a database or dbspace: the main IQ store dbspace, temporary store dbspaces, and Message Log are restored relative to the location of `db_file` (the catalog store); user-created IQ store dbspaces are restored relative to the directory that holds the main IQ dbspace.

  Do not use the RENAME clause to move the `SYSTEM` dbspace, which holds the catalog store. To move the catalog store, and any files created relative to it and not specified in a RENAME clause, specify a new location in the *db_file* parameter.
- **VERIFY [COMPATIBLE –** directs the server to validate the specified SAP Sybase IQ database backup archives for a full, incremental, incremental since full, or virtual backup. The backup must be SAP Sybase IQ version 12.6 or later. The verification process checks the specified archives for the same errors a restore process checks, but performs no write operations. All status messages and detected errors are written to the server log file.

  You cannot use the RENAME clause with the VERIFY clause; an error is reported.

  The backup verification process can run on a different host than the database host. You must have the BACKUP DATABASE system privilege to run **RESTORE DATABASE VERIFY**.

  If the COMPATIBLE clause is specified with VERIFY, the compatibility of an incremental archive is checked with the existing database files. If the database files do not exist on the system on which **RESTORE DATABASE…VERIFY COMPATIBLE** is invoked, an error is returned. If COMPATIBLE is specified while verifying a full backup, the keyword is ignored; no compatibility checks need to be made while restoring a full backup.

  You must have the database and log files (`.db` and `.log`) to validate the backup of a read-only dbspace within a full backup. If you do not have these files, validate the entire backup by running **RESTORE DATABASE…VERIFY** without the READONLY *dbspace* clause.

  Note: The verification of a backup archive is different than the database consistency checker (DBCC) verify mode (`sp_iqcheckdb 'verify...'`). **RESTORE DATABASE VERIFY** validates the consistency of the backup archive to be sure it can be restored, whereas DBCC validates the consistency of the database data.

Run `sp_iqcheckdb 'verify...'` before taking a backup. If an inconsistent database is backed up, then restored from the same backup archive, the data continues to be in an inconsistent state, even if **RESTORE DATABASE VERIFY** reports a successful validation.

Examples

(back to top) on page 73

- **Example 1 –** this UNIX example restores the `iqdemo` database from tape devices `/dev/rmt/0` and `/dev/rmt/2` on a Sun Solaris platform. On Solaris, a restore from tape must specify the use of the rewinding device. Therefore, do not include the letter 'n' after the device name, which specifies "no rewind on close." To specify this feature with **RESTORE DATABASE**, use the naming convention appropriate for your UNIX platform. (Windows does not support this feature.)

```
RESTORE DATABASE 'iqdemo'
FROM '/dev/rmt/0'
FROM '/dev/rmt/2'
```

- **Example 2 –** restore an encrypted database named `marvin` that was encrypted with the key *is!seCret*:

```
RESTORE DATABASE 'marvin'
FROM 'marvin_bkup_file1'
FROM 'marvin_bkup_file2'
FROM 'marvin_bkup_file3'
KEY 'is!seCret'
```

- **Example 3 –** this example shows the syntax of a **BACKUP DATABASE** statement and two possible **RESTORE DATABASE** statements. (This example uses objects in the `iqdemo` database for illustration purposes. Note that `iqdemo` includes a sample user dbspace named `iq_main` that may not be present in your database.)

  Given this **BACKUP DATABASE** statement:

```
BACKUP DATABASE READONLY DBSPACES iq_main
TO '/system1/IQ16/demo/backup/iqmain'
```

  The dbspace `iq_main` can be restored using either of these **RESTORE DATABASE** statements:

```
RESTORE DATABASE 'iqdemo' READONLY DBSPACES iq_main
FROM '/system1/IQ16/demo/backup/iqmain'
```

  or

```
RESTORE DATABASE 'iqdemo'
FROM '/system1/IQ16/demo/backup/iqmain'
```

  A selective backup backs up either all READWRITE dbspaces or specific read-only dbspaces or dbfiles. Selective backups are a subtype of either full or incremental backups.

Notes:

- You can take a READONLY selective backup and restore all objects from this backup (as in the second example above).
- You can take an all-inclusive backup and restore read-only files and dbspaces selectively.
- You can take a READONLY selective backup of multiple read-only files and dbspaces and restore a subset of read-only files and dbspaces selectively. See *Permissions*.
- You can restore the read-only backup, only if the read-only files have not changed since the backup. Once the dbspace is made read-write again, the read-only backup is invalid, unless you restore the entire read-write portion of the database back to the point at which the read-only dbspace was read-only.
- Decide which backup subtype to use (either selective or non-selective) and use it consistently. If you must switch from a non-selective to a selective backup, or vice versa, always take a non-selective full backup before switching to the new subtype, to ensure that you have all changes.

- **Example 4 –** syntax to validate the database archives using the **VERIFY** clause, without performing any write operations:

```
RESTORE DATABASE <database_name.db>
FROM '/sys1/dump/dmp1'
FROM '/sys1/dump/dmp2'
VERIFY
```

When you use validate, specify a different database name to avoid `Database name not unique` errors. If the original database is `iqdemo.db`, for example, use `iq_demo_new.db` instead:

```
RESTORE DATABASE iqdemo_new.db FROM iqdemo.bkp VERIFY
```

Usage

(back to top) on page 73

The **RESTORE DATABASE** command requires exclusive access by a user with the SERVER OPERATOR system privilege to the database. This exclusive access is achieved by setting the **-gd** switch to `DBA`, which is the default when you start the server engine.

Issue the **RESTORE DATABASE** command before you start the database (you must be connected to the `utility_db` database). Once you finish specifying **RESTORE DATABASE** commands for the type of backup, that database is ready to be used. The database is left in the state that existed at the end of the first implicit **CHECKPOINT** of the last backup you restored. You can now specify a **START DATABASE** to allow other users to access the restored database.

The maximum size for a complete **RESTORE DATABASE** command, including all clauses, is 32KB.

When restoring to a raw device, make sure the device is large enough to hold the dbspace you are restoring. SAP Sybase IQ **RESTORE DATABASE** checks the raw device size and returns an error, if the raw device is not large enough to restore the dbspace.

BACKUP DATABASE allows you to specify full or incremental backups. There are two kinds of incremental backups. INCREMENTAL backs up only those blocks that have changed and committed since the last backup of any type (incremental or full). INCREMENTAL SINCE FULL backs up all the blocks that have changed since the last full backup. If a restore of a full backup is followed by one or more incremental backups (of either type), no modifications to the database are allowed between successive **RESTORE DATABASE** commands. This rule prevents a restore from incremental backups on a database in need of crash recovery, or one that has been modified. You can still overwrite such a database with a restore from a full backup.

Before starting a full restore, you must delete two files: the catalog store file (default name `dbname.db`) and the transaction log file (default name `dbname.log`).

If you restore an incremental backup, **RESTORE DATABASE** ensures that backup media sets are accessed in the proper order. This order restores the last full backup tape set first, then the first incremental backup tape set, then the next most recent set, and so forth, until the most recent incremental backup tape set. If a user with the SERVER OPERATOR system privilege produced an INCREMENTAL SINCE FULL backup, only the full backup tape set and the most recent INCREMENTAL SINCE FULL backup tape set is required; however, if there is an INCREMENTAL backup made since the INCREMENTAL SINCE FULL backup, it also must be applied.

SAP Sybase IQ ensures that the restoration order is appropriate, or it displays an error. Any other errors that occur during the restore results in the database being marked corrupt and unusable. To clean up a corrupt database, do a restore from a full backup, followed by any additional incremental backups. Since the corruption probably happened with one of those backups, you might need to ignore a later backup set and use an earlier set.

To restore read-only files or dbspaces from an archive backup, the database may be running and the administrator may connect to the database when issuing the **RESTORE DATABASE** statement. The read-only file pathname need not match the names in the backup, if they otherwise match the database system table information.

The database must not be running to restore a FULL, INCREMENTAL SINCE FULL, or INCREMENTAL restore of either a READWRITE FILES ONLY or an all files backup. The database may or may not be running to restore a backup of read-only files. When restoring specific files in a read-only dbspace, the dbspace must be offline. When restoring read-only files in a read-write dbspace, the dbspace can be online or offline. The restore closes the read-only files, restores the files, and reopens those files at the end of the restore.

You can use selective restore to restore a read-only dbspace, as long as the dbspace is still in the same read-only state.

Other **RESTORE DATABASE** issues:

- **RESTORE DATABASE** to disk does not support raw devices as archival devices.
- SAP Sybase IQ does not rewind tapes before using them; on rewinding tape devices, it does rewind tapes after using them. You must position each tape to the start of the SAP Sybase IQ data before starting the restore.
- During backup and restore operations, if SAP Sybase IQ cannot open the archive device (for example, when it needs the media loaded) and the ATTENDED option is ON, it waits for ten seconds for you to put the next tape in the drive, and then tries again. It continues these attempts indefinitely until either it is successful or the operation is terminated with Ctrl+C.
- If you press Ctrl+C, **RESTORE DATABASE** fails and returns the database to its state before the restoration began.
- If disk striping is used, the striped disks are treated as a single device.
- The `file_name` column in the `SYSFILE` system table for the `SYSTEM` dbspace is not updated during a restore. For the `SYSTEM` dbspace, the `file_name` column always reflects the name when the database was created. The file name of the `SYSTEM` dbspace is the name of the database file.

Standards

(back to top) on page 73

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- SAP Sybase Database product—Not supported by Adaptive Server.

Permissions

(back to top) on page 73

The permissions required to execute this statement are set using the **-gu** server command line option, as follows:

- **NONE –** No user can issue this statement.
- **DBA –** Requires the SERVER OPERATOR system privilege.
- **UTILITY_DB –** Only those users who can connect to the `utility_db` database can issue this statement.

sp_iqcheckdb Procedure

Checks validity of the current database. Optionally corrects allocation problems for dbspaces or databases. **sp_iqcheckdb** does not check a partitioned table if partitioned data exists on offline dbspaces.

sp_iqcheckdb reads all storage in the database. On successful completion, the database free list (an internal allocation map) is updated to reflect the true storage allocation for the database. **sp_iqcheckdb** then generates a report listing the actions it has performed.

If an error is found, **sp_iqcheckdb** reports the name of the object and the type of error. **sp_iqcheckdb** does not update the free list if errors are detected.

sp_iqcheckdb also allows you to check the consistency of a specified table, index, index type, or the entire database.

Note: sp_iqcheckdb is the user interface to the SAP Sybase IQ database consistency checker (DBCC) and is sometimes referred to as **DBCC**.

Syntax

```
sp_iqcheckdb 'mode target [ … ] [ resources resource-percent ]'

mode :
    { allocation
    | check
    | verify }
    | dropleaks

target :
    [ indextype index-type […] ]   database
    | database resetclocks
    | { [ indextype index-type ] […] table table-name [ partition partition-
name ] […]
    | index index-name
    |   […] dbspace   dbspace-name}
    | cache main-cache-name
```

There are three modes for checking database consistency, and one for resetting allocation maps. If mode and target are not both specified in the parameter string, SAP Sybase IQ returns the error message:

```
At least one mode and target must be specified to DBCC.
```

Parameter

- **database –** If the target is a database, all dbspaces must be online.
- **index-type –** One of the following index types: **FP**, **CMP**, **LF**, **HG**, **HNG**, **WD**, **DATE**, **TIME**, **DTTM**, **TEXT**.

If the specified *index-type* does not exist in the target, an error message is returned. If multiple index types are specified and the target contains only some of these index types, the existing index types are processed by **sp_iqcheckdb**.

- **index-name –** May contain owner and table qualifiers: `[[owner.]table-name.]index-name`

  If *owner* is not specified, current user and database owner (`dbo`) are substituted in that order. If *table* is not specified, *index-name* must be unique.
- **table-name –** May contain an owner qualifier: `[owner.]table-name`

  If *owner* is not specified, current user and database owner (`dbo`) are substituted in that order. *table-name* cannot be a temporary or pre-join table.

Note: If either the table name or the index name contains spaces, enclose the *table-name* or *index-name* parameter in double quotation marks:

```
sp_iqcheckdb 'check index "dbo.sstab.i2" resources 75'
```

- **partition-name –** The *partition-name* parameter contains no qualifiers. If it contains spaces, enclose it in double quotation marks.

  The partition filter causes **sp_iqcheckdb** to examine a subset of the corresponding table's rows that belong to that partition. A partition filter on a table and table target without the partition filter are semantically equivalent when the table has only one partition.
- **dbspace-name –** The *dbspace-name* parameter contains no qualifiers. If it contains spaces, enclose it in double quotation marks.

  The dbspace target examines a subset of the database's pages that belong to that dbspace. The dbspace must be online. The dbspace and database target are semantically equivalent when the table has only one dbspace.
- **resource-percent –** The input parameter *resource-percent* must be an integer greater than zero. The resources percentage allows you to limit the CPU utilization of the database consistency checker by controlling the number of threads with respect to the number of CPUs. If *resource-percent* = 100 (the default value), then one thread is created per CPU. If *resource-percent* > 100, then there are more threads than CPUs, which might increase performance for some machine configurations. The minimum number of threads is one.
- **main-cache-name –** The cache target compares pages in the main cache dbspace against the original pages in the IQ main store.

Note: The **sp_iqcheckdb** parameter string must be enclosed in single quotes and cannot be greater than 255 bytes in length.

Allocation problems can be repaired in dropleaks mode.

Applies to
Simplex and multiplex.

Privileges

You must have EXECUTE privilege on the system procedure, as well as the ALTER DATABASE system privilege.

Remarks

sp_iqcheckdb checks the allocation of every block in the database and saves the information in the current session until the next **sp_iqdbstatistics** procedure is issued. **sp_iqdbstatistics** displays the latest result from the most recent execution of **sp_iqcheckdb**.

sp_iqcheckdb can perform several different functions, depending on the parameters specified.

Mode	Description
Allocation	Checks allocation with blockmap information for the entire database, a specific index, a specific index type, a specific partition, specific table, or a specific dbspace. Does not check index consistency. Detects duplicate blocks (blocks for which two or more objects claim ownership) or extra blocks (unallocated blocks owned by an object). Detects leaked blocks (allocated blocks unclaimed by any object in the specified target) for database or dbspace targets. When the target is a partitioned table, **allocation mode**: • Checks metadata of all the table's partition allocation bitmaps • Checks metadata of the tables allocation bitmap • Verifies that blockmap entries are consistent with the table's allocation bitmap • Verifies that none of the table's partition allocation bitmaps overlap • Checks that rows defined in the table's partition allocation bitmaps form a superset of the table's existence bitmap • Checks that rows defined in the table's partition allocation bitmaps form a superset of the table's allocation bitmap • Verifies that the main cache pages are consistent with the IQ main store pages. **Note: sp_iqcheckdb** cannot check all allocation problems if you specify the name of a single index, index type, or table in the input parameter string. Run in allocation mode: • To detect duplicate or unowned blocks (use database or specific tables or indexes as the target) • If you encounter page header errors The DBCC option **resetclocks** is used only with allocation mode. **resetclocks** is used with forced recovery to convert a multiplex secondary server to a coordinator. For information on multiplex capability, see *Administration: Multiplex*. **resetclocks** corrects the values of internal database versioning clocks, in the event that these clocks are behind. Do not use the **resetclocks** option for any other purpose, unless you contact SAP Sybase IQ Technical Support. The **resetclocks** option must be run in single-user mode and is allowed only with the DBCC statement **allocation database**. The syntax of **resetclocks** is: `sp_iqcheckdb 'allocation database resetclocks'`

Mode	Description
Check	Verifies that all database pages can be read for the entire database, main cache, specific index, specific index type, specific table, specific partition, or specific dbspace. If the table is partitioned, then check mode will check the table's partition allocation bitmaps. Run in check mode if metadata, null count, or distinct count errors are returned when running a query.
Verify	Verifies the contents of non-FP indexes with their corresponding FP indexes for the entire database, main cache, a specific index, a specific index type, specific table, specific partition, or specific dbspace. If the specified target contains all data pages for the FP and corresponding non-FP indexes, then verify mode detects the following inconsistencies: • **Missing key –** a key that exists in the FP but not in the non-FP index. • **Extra key –** a key that exists in the non-FP index but not in the FP index. • **Missing row –** a row that exists in the FP but not in the non-FP index. • **Extra row –** a row that exists in the non-FP index but not in the FP index. If the specified target contains only a subset of the FP pages, then verify mode can detect only the following inconsistencies: • Missing key • Missing row If the target is a partitioned table, then verify mode also verifies that each row in the table or table partition has been assigned to the correct partition. Run in verify mode if metadata, null count, or distinct count errors are returned when running a query. **Note: sp_iqcheckdb** does not check referential integrity or repair referential integrity violations.
Dropleaks	When the SAP Sybase IQ server runs in single-node mode, you can use dropleaks mode with either a database or dbspace target to reset the allocation map for the entire database or specified dbspace targets. If the target is a dbspace, then the dropleaks operation must also prevent read-write operations on the named dbspace. All dbspaces in the database or dbspace list must be online. On a multiplex coordinator node, dropleaks mode also detects leaked blocks, duplicate blocks, or extra blocks across the multiplex.

DBCC Performance:

The execution time of DBCC varies, depending on the size of the database for an entire database check, the number of tables or indexes specified, and the size of the machine.

Checking only a subset of the database (that is, only specified tables, indexes, or index types) requires less time than checking an entire database.

The processing time of **sp_iqcheckdb** dropleaks mode depends on the number of dbspace targets.

This table summarizes the actions and output of the four **sp_iqcheckdb** modes.

Table 14. Actions and Output of sp_iqcheckdb Modes

Mode	Errors Detected	Output	Speed
Allocation	Allocation errors	Allocation statistics only	4TB per hour
Check	Allocation errors Most index errors	All available statistics	60GB per hour
Verify	Allocation errors All index errors	All available statistics	15GB per hour
Dropleaks	Allocation errors	Allocation statistics only	4TB per hour

Output:

Depending on the execution mode, **sp_iqcheckdb** output includes summary results, errors, informational statistics, and repair statistics. The output may contain as many as three results sets, if you specify multiple modes in a single session. Error statistics are indicated by asterisks (*****), and appear only if errors are detected.

The output of **sp_iqcheckdb** is also copied to the SAP Sybase IQ message file `.iqmsg`. If the **DBCC_LOG_PROGRESS** option is ON, **sp_iqcheckdb** sends progress messages to the IQ message file, allowing the user to follow the progress of the DBCC operation as it executes.

Example

Check the allocation for the entire database:

```
sp_iqcheckdb 'allocation database'
```

Perform a detailed check on indexes `i1`, `i2`, and `dbo.t1.i3`. If you do not specify a new mode, **sp_iqcheckdb** applies the same mode to the remaining targets, as shown in the following command:

```
sp_iqcheckdb 'verify index i1 index i2 index dbo.t1.i3'
```

You can combine all modes and run multiple checks on a database in a single session. Perform a quick check of partition `p1` in table `t2`, a detailed check of index `i1`, and allocation checking for the entire database using half of the CPUs:

```
sp_iqcheckdb 'check table t2 partition p1 verify index i1
allocation database resources 50'
```

Check all indexes of the type **FP** in the database:

```
sp_iqcheckdb 'check indextype FP database'
```

Verify the **FP** and **HG** indexes in the table `t1` and the **LF** indexes in the table `t2`:

```
sp_iqcheckdb 'verify indextype FP indextype HG table t1 indextype LF
table t2'
```

Check for LVC cell inconsistencies:

```
sp_iqcheckdb 'check index EFG2JKL.ASIQ_IDX_T208_C504_FP'
-----------------------------------
Index Statistics:
** Inconsistent Index: abcd.EFG2JKL.ASIQ_IDX_T208_C504_FP ****** FP
Indexes Checked: 1
** Unowned LVC Cells: 212 ******
```

The **sp_iqcheckdb** LVC cells messages include:

- Unowned LVC cells
- Duplicate LVC cell rows
- Unallocated LVC cell rows

These messages indicate inconsistencies with a `VARCHAR`, `VARBINARY`, `LONG BINARY` (`BLOB`), or `LONG VARCHAR` (`CLOB`) column. Unowned LVC cells represent a small amount of unusable disk space and can safely be ignored. Duplicate and Unallocated LVC cells are serious errors that can be resolved only by dropping the damaged columns.

To drop a damaged column, create a new column from a copy of the old column, then drop the original column and rename the new column to the old column.

Note: LVC is a `VARCHAR` or `VARBINARY` column with a width greater than 255. `LONG BINARY` (`BLOB`) and `LONG VARCHAR` (`CLOB`) also use LVC.

Output Example:
Run **sp_iqcheckdb 'allocation database'**:

```
======================================================================
DBCC Allocation Mode Report
======================================================================
   DBCC Status                                    No Errors Detected
======================================================================
Allocation Summary
======================================================================

   Blocks Total                                    25600
   Blocks in Current Version                       5917
   Blocks in All Versions                          5917
   Blocks in Use                                   5917
   % Blocks in Use                                 23
======================================================================
Allocation Statistics
======================================================================
   Marked Logical Blocks                           8320
   Marked Physical Blocks                          5917
```

```
   Marked Pages                                    520
   Blocks in Freelist                              2071196
   Imaginary Blocks                                2014079
   Highest PBN in Use                              1049285
   Total Free Blocks                               19683
   Usable Free Blocks                              19382
   % Total Space Fragmented                        1
   % Free Space Fragmented                         1
   Max Blocks Per Page                             16
   1  Block Page Count                             165
   3  Block Page Count                             200
   4  Block Page Count                             1
   10 Block Page Count                             1
   16 Block Page Count                             153
   2  Block Hole Count                             1
   3  Block Hole Count                             19
   6  Block Hole Count                             12
   7  Block Hole Count                             1
   10 Block Hole Count                             1
   15 Block Hole Count                             1
   16 Block Hole Count                             1220

Partition Summary
   Database Objects Checked                        2
   Blockmap Identity Count                         2
   Bitmap Count                                    2
=================================================================
Connection Statistics
=================================================================
   Sort Records                                    3260
   Sort Sets                                       2
=================================================================

DBCC Info
=================================================================
   DBCC Work units Dispatched                      197
   DBCC Work units Completed                       197
   DBCC Buffer Quota                               255
   DBCC Per-Thread Buffer Quota                    255
   Max Blockmap ID found                           200
   Max Transaction ID found                        404
```

Note: The report may indicate leaked space. Leaked space is a block that is allocated according to the database free list (an internal allocation map), but DBCC finds that the block is not part of any database object.

Index

A

B

C

D

S

T

V

W