

ADMINISTRATION GUIDE | PUBLIC
SAP IQ 16.1 SP 01
Document Version: 1.0.0 – 2018-12-20

Administration: Backup, Restore, and Data Recovery

© 2019 SAP SE or an SAP affiliate company. All rights reserved.

THE BEST RUN SAP

Content

1 SAP IQ Backup, Restore, and Data Recovery. 4

2 Data Backup and Recovery. 5

2.1 Database Backup. 6

2.2 Database Validation. 8

2.3 Performance Options. 9

2.4 Archive Devices. 11

Tape Backups. 11

Disk Backups. 14

2.5 Queries, Utilities, and Procedures. 16

2.6 Backup Scenarios. 18

Routine Backups. 19

Cache Dbspace Backup. 21

System-Level Backups. 22

Virtual Backups. 25

2.7 Previous Backups. 29

2.8 Backup Log. 30

3 Restoring Database Backups. 33

3.1 Database Restore. 34

3.2 Verifying Database Backups. 37

3.3 Displaying Header Information. 39

3.4 Restoring to Raw Devices. 40

3.5 Restoring Cache Dbspaces. 42

3.6 Moving Database Files. 43

3.7 Restoring Multiplex Stores. 45

Restoring Multiplex Stores to a Different Location. 46

Restoring Multiplex Stores to the Same Location. 48

Restoring Read-only Backups for a Coordinator. 50

3.8 Error Recovery. 51

4 Point-in-Time Recovery Operations. 52

4.1 Point-in-Time Recovery. 52

4.2 Alternate Database Timelines 59

4.3 Retrieving Timeline Values. 62

5 Shared-Nothing Multiplex Backup and Recovery. 66

5.1 Shared-Nothing Multiplex Full Backup. 67

5.2 Shared-Nothing Multiplex Full Restore . . . 67
5.3 Shared-Nothing Multiplex Incremental Backup. . . . 68
5.4 Shared-Nothing Multiplex Incremental Restore with Configuration Changes. . . . 69
5.5 Shared-Nothing Multiplex Incremental Restore to a Different Location. . . . 70

6 System Recovery and Database Repair. . . . 73
6.1 Recovery and Repair Overview. . . . 73
6.2 Normal Recovery. . . . 74
6.3 Database Verification. . . . 74
Database Verification Using the sp_iqcheckdb Stored Procedure. . . . 75
Analyzing sp_iqcheckdb Output. . . . 78
Resource Issues Running sp_iqcheckdb. . . . 81
6.4 Database Repair. . . . 82
Analysis of Index Errors. . . . 82
Index Error Repair. . . . 85
Analysis of Allocation Problems. . . . 85
Repairing Allocation Problems using DBCC. . . . 87
6.5 Forced Recovery Mode. . . . 89
Before Forced Recovery. . . . 89
Starting a Server in Forced Recovery Mode. . . . 90
Recovering Leaked Space. . . . 92
6.6 Problems Reported by DBCC. . . . 93
Index Problems that DBCC Cannot Repair. . . . 94
Dropping Inconsistent Indexes, Tables, or Columns. . . . 95
6.7 DBCC Error Messages. . . . 96

7 Backup Reference. . . . 99
7.1 BACKUP DATABASE Statement. . . . 99
7.2 RESTORE DATABASE Statement. . . . 105
7.3 sp_iqcheckdb Procedure. . . . 116

1 SAP IQ Backup, Restore, and Data Recovery

Data backup, restore, and recovery procedures.

2 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.

In this section:

Database Backup [page 6]
The BACKUP DATABASE command backs up an SAP IQ database to one or more archive devices.

Database Validation [page 8]
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.

Performance Options [page 9]
`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.

Archive Devices [page 11]
Specific information about writing to different archive devices, including tape, disk, read-only hardware, and third-party products.

Queries, Utilities, and Procedures [page 16]
Queries, utilities, and procedures return information about the tablespaces, dbspaces, and dbfiles in your database.

Backup Scenarios [page 18]
Background information and code samples for routine, system-level, and virtual backups.

Previous Backups [page 29]
Views, utilities, and procedures return details about previous backups.

Backup Log [page 30]
SAP IQ logs events that occur during backups and restores to `.backup.syb`.

2.1 Database Backup

The BACKUP DATABASE command backs up an SAP 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.

The endian format and platform on the system where you back up a database must match the endian format and platform where you restore it.

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 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 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.

> **i** Note
>
> See *BACKUP DATABASE* in *SAP IQ SQL Reference* for parameter descriptions, usage, and required permissions.

Parent topic: Data Backup and Recovery [page 5]

Related Information

Database Validation [page 8]
Performance Options [page 9]
Archive Devices [page 11]
Queries, Utilities, and Procedures [page 16]
Backup Scenarios [page 18]

Previous Backups [page 29]
Backup Log [page 30]
Point-in-Time Recovery Operations [page 52]

2.2 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'`

Parent topic: Data Backup and Recovery [page 5]

Related Information

Database Backup [page 6]
Performance Options [page 9]
Archive Devices [page 11]

Queries, Utilities, and Procedures [page 16]
Backup Scenarios [page 18]
Previous Backups [page 29]
Backup Log [page 30]

2.3 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 IQ page size of 128 KB 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 backup. 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	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 specify a string as long as 32 KB 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 IQ needs to manage your database. However, you can create non-IQ tables in the catalog store.

To improve performance, keep any non-SAP IQ data in a separate SAP SQL Anywhere-only database, instead of 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.

Parent topic: Data Backup and Recovery [page 5]

Related Information

Database Backup [page 6]
Database Validation [page 8]
Archive Devices [page 11]
Queries, Utilities, and Procedures [page 16]

Backup Scenarios [page 18]
Previous Backups [page 29]
Backup Log [page 30]

2.4 Archive Devices

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

In this section:

Tape Backups [page 11]
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 IQ data in the catalog store.

Disk Backups [page 14]
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.

Parent topic: Data Backup and Recovery [page 5]

Related Information

Database Backup [page 6]
Database Validation [page 8]
Performance Options [page 9]
Queries, Utilities, and Procedures [page 16]
Backup Scenarios [page 18]
Previous Backups [page 29]
Backup Log [page 30]

2.4.1 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 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/0<n>`
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/0m<n>`

> **i Note**
>
> See your platform documentation for specific instructions and options.

Tape Devices on Windows

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

Because SAP 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.

> **i** Note
>
> SAP 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 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
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 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 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 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.

Parent topic: Archive Devices [page 11]

Related Information

Disk Backups [page 14]

2.4.2 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	2 GB
Windows	1.5 GB

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 (8 KB 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 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.

Parent topic: Archive Devices [page 11]

Related Information

Tape Backups [page 11]

2.5 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/IQ/demo/iqdemo.db   system                  1 (NULL)           0
temporary                15 /temp/sqla0000.tmp                temporary              1 (NULL)          15
IQ_SYSTEM_LOG         32702  pitrLog/PITR                     IQ_SYSTEM_LOG          2  32702           0
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 *SYSDBFILE System View* in *SAP IQ SQL Reference*.

db_backupheader Utility

The `db_backupheader` utility reads the first backup archive, and 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  IsName
---------------- ------------ -------- ------ ----- --------- ------- -------- ----------- ---------- ----------
---------------- --------  ------
iq_main           MAIN         T        T      26    100M      200M           1           1 T          1K         1H,
3254A            N        (NULL)
IQ_SYSTEM_LOG     PITR         T        T      0     0B        0B             1           1 F          0B
1H                 N        (NULL)
IQ_SYSTEM_MAIN    MAIN         T        T      22    100M      200M           1           1 T          1K         1H,
2528F,32D,128M   N        (NULL)
IQ_SYSTEM_TEMP    TEMPORARY    T        T      3     25M       200M           1           1 T          1K         1H,
64F,16A          N        (NULL)
```

See *sp_iqdbspace Procedure* in *SAP IQ SQL Reference* 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  servername  morrorLogicalFileName
--------------- ---------------- -------------- ----------  ------ ------ ----- ---------- ------- ----------
----------------- --------- -------- --------- ----------  ---------------------
IQ_SYSTEM_MAIN  IQ_SYSTEM_MAIN   iqdemo.iq      MAIN        RW     T      22    100M       200M    1K         1H,
2528F,32D,128M          1    12800  N        (NULL)      (NULL)
iq_main         iq_main          iqdemo_main.iq MAIN        RW     T      26    100M       200M    1K         1H,
3254A             1045440  1058239  N        (NULL)      (NULL)
IQ_SYSTEM_LOG   IQ_SYSTEM_LOG    pitrLog/PITR   PITR        RW    T      0     0B         0B      0B
1H                       1        1  N        (NULL)      (NULL)
IQ_SYSTEM_TEMP  IQ_SYSTEM_TEMP   iqdemo.iqtmp   TEMPORARY   RW     T      3     25M        200M    1K         1H,
64F,16A                 1     3200  N        (NULL)      (NULL)
```

See *sp_iqfile Procedure* in *SAP IQ SQL Reference*.

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'
```

> **i** 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.

Parent topic: Data Backup and Recovery [page 5]

Related Information

Database Backup [page 6]
Database Validation [page 8]
Performance Options [page 9]
Archive Devices [page 11]
Backup Scenarios [page 18]
Previous Backups [page 29]
Backup Log [page 30]
Cache Dbspace Backup [page 21]

2.6 Backup Scenarios

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

In this section:

Routine Backups [page 19]
: Code samples, background information, and tips for standard backup options.

Cache Dbspace Backup [page 21]
: If your installation uses the cache dbspace for direct-attached storage, follow best practices to avoid errors during backup.

System-Level Backups [page 22]
: 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.

Virtual Backups [page 25]
: Virtual backups perform a full backup of the catalog store and select SAP 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.

Parent topic: Data Backup and Recovery [page 5]

Related Information

Database Backup [page 6]
Database Validation [page 8]
Performance Options [page 9]
Archive Devices [page 11]
Queries, Utilities, and Procedures [page 16]
Previous Backups [page 29]
Backup Log [page 30]

2.6.1 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 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'
```

Parent topic: Backup Scenarios [page 18]

Related Information

Cache Dbspace Backup [page 21]
System-Level Backups [page 22]
Virtual Backups [page 25]

2.6.2 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 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.

Parent topic: Backup Scenarios [page 18]

Related Information

Routine Backups [page 19]
System-Level Backups [page 22]
Virtual Backups [page 25]

2.6.3 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 [page 22]
   Shut down your SAP IQ database before performing a system-level backup.
2. Back Up the Correct Files [page 23]
   Back up required files and optional files.
3. Restoring System-Level Backups [page 24]
   Shut down the database server. On a multiplex, shut down all the secondary servers as well as the coordinator.

Parent topic: Backup Scenarios [page 18]

Related Information

Routine Backups [page 19]
Cache Dbspace Backup [page 21]
Virtual Backups [page 25]

2.6.3.1 Shut Down the Database

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

Ensure that no one starts the SAP 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.

Parent topic: System-Level Backups [page 22]

Next: Back Up the Correct Files [page 23]

2.6.3.2 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 IQ can reconstruct any temporary dbspace as long as 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

Even though these files are not required for a restore, back up the ASCII message files such as `<dbname>.iqmsg`, `*.srvlog`, and `*.stderr` files in the `$IQDIR16/logfiles/` directory (for UNIX; `%IQDIR16%\logfiles\` for Windows). 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 you can access all messages 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 128 GB, which is sufficient in most cases.

> **i** 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.

Parent topic: System-Level Backups [page 22]

Previous: Shut Down the Database [page 22]

Next: Restoring System-Level Backups [page 24]

2.6.3.3 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 perform a restoration:

- Review the table of contents of the backup to ensure that all required SAP 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.

Parent topic: System-Level Backups [page 22]

Previous: Back Up the Correct Files [page 23]

2.6.4 Virtual Backups

Virtual backups perform a full backup of the catalog store and select SAP 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.

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.

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 you 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.

In this section:

Performing Full Virtual Decoupled Backups [page 27]
Back up the database with a full virtual decoupled backup.

Restoring Virtual Decoupled Backups without DAS Dbspaces [page 27]
Restore databases without direct-attached storage (DAS) dbspaces from virtual decoupled backups.

Restoring Virtual Decoupled Backups with DAS Dbspaces [page 28]
Restore databases with direct-attached storage (DAS) dbspaces from virtual decoupled backups.

Parent topic: Backup Scenarios [page 18]

Related Information

Routine Backups [page 19]
Cache Dbspace Backup [page 21]
System-Level Backups [page 22]

2.6.4.1 Performing Full Virtual Decoupled Backups

Back up the database with a full virtual decoupled backup.

Procedure

1. Perform a full decoupled virtual data backup:

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

2. Use a shell command to back up the IQ store:

```
dd if=iqdemo.iq of=iqdemo.iq.copy
```

3. Perform a non-virtual incremental backup:

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

2.6.4.2 Restoring Virtual Decoupled Backups without DAS Dbspaces

Restore databases without direct-attached storage (DAS) dbspaces from virtual decoupled backups.

Procedure

1. Shut down the server.
2. Use a shell command to restore the system-level copy of IQ store:

```
dd if=iqdemo.copy of=iqdemo.iq
```

3. Start the utility database.

4. Restore the SAN virtual full backup:

```
RESTORE DATABASE 'iquser'
  FROM 'iqdemo.full';
```

5. Restore the SAN incremental backup since the full backup:

```
RESTORE DATABASE 'iquser' FROM 'iqdemo.isf';
```

6. Stop the utility database.
7. Start the server. On multiplex servers, perform the following:
   a. Start the coordinator.
   b. Synchronize the secondary nodes.
   c. Start all secondary nodes.

2.6.4.3 Restoring Virtual Decoupled Backups with DAS Dbspaces

Restore databases with direct-attached storage (DAS) dbspaces from virtual decoupled backups.

Procedure

1. Shut down the server.
2. Use a shell command to restore the system-level copy of IQ store:

```
dd if=iqdemo.copy of=iqdemo.iq
```

3. Start the utility database.
4. Restore the SAN virtual full backup:

```
RESTORE DATABASE 'iquser'
  FROM 'iqdemo.full';
```

5. Restore the SAN incremental backup since the full backup:

```
RESTORE DATABASE 'iquser' FROM 'iqdemo.isf';
```

6. Stop the utility database.
7. Start the coordinator.
8. Synchronize, then start all the secondary nodes.
9. Restore the DAS virtual full backup:

```
RESTORE DATABASE 'iquser' FROM 'iqdemo.full' MULTIPLEX;
```

10. Restore the DAS incremental backup since the full backup:

```
RESTORE DATABASE 'iquser' FROM 'iqdemo.isf' MULTIPLEX;
```

2.7 Previous Backups

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

SYSIQBACKUPHISTORY

Returns details about previous backups:

```
select * from SYSIQBACKUPHISTORY
```

SYSIQBACKUPHISTORY is a system view. See *SYSIQBACKUPHISTORY System View* in *SAP IQ SQL Reference* for additional information.

db_backupheader Utility

Returns backup statistics, timeline and log details:

```
db_backupheader [ <path> ] <backup_file>
```

`db_backupheader` is a command line utility. See *db_backupheader Utility* in *SAP IQ Utility Guide* for additional information.

sp_iqbackupsummary Procedure

Returns details about backup operations by `timestamp` or `backup_id`:

```
sp_iqbackupsummary [ <timestamp> or <backup_id> ]
```

`sp_iqbackupsummary` is a stored procedure. See *sp_iqbackupsummary Procedure* in *SAP IQ SQL Reference* for additional information.

sp_iqbackupdetails Procedure

Returns results about backup operations by `backup_id`:

```
 sp_iqbackupdetails '<backup_id>'
```

`sp_iqbackupdetails` is a stored procedure. See *sp_iqbackupdetails Procedure* in *SAP IQ SQL Reference* for additional information.

sp_iqrestoreaction Procedure

Returns actions required to restore the database to a consistent state for a given date:

```
sp_iqrestoreaction ' <timestamp> ' , , [ <backup_id> ]
```

`sp_iqrestoreaction` is a stored procedure. See *sp_iqrestoreaction Procedure* in *SAP IQ SQL Reference* for additional information.

Parent topic: Data Backup and Recovery [page 5]

Related Information

Database Backup [page 6]
Database Validation [page 8]
Performance Options [page 9]
Archive Devices [page 11]
Queries, Utilities, and Procedures [page 16]
Backup Scenarios [page 18]
Backup Log [page 30]

2.8 Backup Log

SAP 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.
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.

> **i** 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.

Parent topic: Data Backup and Recovery [page 5]

Related Information

Database Backup [page 6]
Database Validation [page 8]
Performance Options [page 9]
Archive Devices [page 11]
Queries, Utilities, and Procedures [page 16]
Backup Scenarios [page 18]
Previous Backups [page 29]

3 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.

The endian format and platform of the system where you restore a database must match the endian format and platform where you backed it up.

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

In this section:

Database Restore [page 34]
Interactive SQL supports two `RESTORE DATABASE` syntax variations.

Verifying Database Backups [page 37]
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.

Displaying Header Information [page 39]
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.

Restoring to Raw Devices [page 40]
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.

Restoring Cache Dbspaces [page 42]
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.

Moving Database Files [page 43]
Redirect the catalog, or use the `RENAME` clause to move the database to a new location on the file system.

Restoring Multiplex Stores [page 45]
Run `RESTORE DATABASE` commands only on the coordinator node. You cannot run RESTORE operations against a secondary server. Before you restore, contact Support to verify that a restore operation is necessary.

Error Recovery [page 51]
Solutions to common RESTORE errors.

3.1 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. In a production system, never start the database between incremental restores. Restore all backups in sequence without restarting SAP IQ until you have a fully restored system.

- If the dbspace name contains a file extension such as `.iq` or `.iqtmp`, you must enclose the dbspace name in double quotation marks when specifying the name in a `RESTORE DATABASE` command `RENAME` clause. For example:

```
RENAME temp1 TO '/work/temp1_res.iqtmp.iqtmp'
DBSPACENAME "temp1_res.iqtmp"
```

Or:

```
RENAME "test_prod2.iq" TO '/test/test_prod2.iq'
```

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. **i Note** If a transaction log with the same name already exists in the target location, RESTORE DATABASE generates an exception and does not restore your files. Fix the problem by using one of these two workarounds: • Rename the transaction log file of the source database using the `dblog` database administration utility. • Rename the transaction log file in the new location using the `RENAME` clause of the `RESTORE DATABASE` statement.
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 do not need to 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 require 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. Shut down the database server.
2. Start a server that you can use to connect to the utility database:

   ```
   start_iq -su <password> -gd DBA -gm 1 -n my_server
   ```
3. Start dbisql and connect to the utility database:

   ```
   dbisql -c "UID=DBA;PWD=<password>;DBN=utility_db"
   ```
4. Run all `RESTORE DATABASE` commands from dbisql. On Windows, you need not redouble the backslashes in path names for tape devices for restore, as you did for `BACKUP DATABASE`.
5. Shut down the server and utility database.
6. Start the database server normally.

When SAP 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 simply run `sp_iqcheckdb` after you restore the last incremental backup.

Parent topic: Restoring Database Backups [page 33]

Related Information

Verifying Database Backups [page 37]
Displaying Header Information [page 39]
Restoring to Raw Devices [page 40]
Restoring Cache Dbspaces [page 42]
Moving Database Files [page 43]
Restoring Multiplex Stores [page 45]
Error Recovery [page 51]

3.2 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 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 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.

> **i 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.

Parent topic: Restoring Database Backups [page 33]

Related Information

Database Restore [page 34]
Displaying Header Information [page 39]
Restoring to Raw Devices [page 40]
Restoring Cache Dbspaces [page 42]
Moving Database Files [page 43]
Restoring Multiplex Stores [page 45]
Error Recovery [page 51]
RESTORE DATABASE Statement [page 105]

3.3 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
```

It then 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,
      ''
```

> **i 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.

Parent topic: Restoring Database Backups [page 33]

Related Information

Database Restore [page 34]
Verifying Database Backups [page 37]
Restoring to Raw Devices [page 40]
Restoring Cache Dbspaces [page 42]
Moving Database Files [page 43]
Restoring Multiplex Stores [page 45]
Error Recovery [page 51]

3.4 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.

Column Name	Description
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.

Parent topic: Restoring Database Backups [page 33]

Related Information

Database Restore [page 34]
Verifying Database Backups [page 37]
Displaying Header Information [page 39]
Restoring Cache Dbspaces [page 42]
Moving Database Files [page 43]
Restoring Multiplex Stores [page 45]

Error Recovery [page 51]

3.5 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.

Parent topic: Restoring Database Backups [page 33]

Related Information

Database Restore [page 34]
Verifying Database Backups [page 37]
Displaying Header Information [page 39]
Restoring to Raw Devices [page 40]
Moving Database Files [page 43]
Restoring Multiplex Stores [page 45]
Error Recovery [page 51]
Restoring Multiplex Stores to a Different Location [page 46]
Restoring Multiplex Stores to the Same Location [page 48]

3.6 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.

> **i** 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 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 *dblog Database Administration Utility* in the *SAP IQ Utility Guide*.

Parent topic: Restoring Database Backups [page 33]

Related Information

Database Restore [page 34]
Verifying Database Backups [page 37]
Displaying Header Information [page 39]

Restoring to Raw Devices [page 40]
Restoring Cache Dbspaces [page 42]
Restoring Multiplex Stores [page 45]
Error Recovery [page 51]

3.7 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 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.

> **i** Note
>
> A shared-nothing multiplex uses a different restore procedure than a multiplex on a shared storage area network. If your implementation uses DAS dbfiles, see *Shared Nothing Multiplex Backup and Recovery* .

In this section:

Restoring Multiplex Stores to a Different Location [page 46]
Restore operations vary, depending on the location from which you restore.

Restoring Multiplex Stores to the Same Location [page 48]
Restore operations vary depending on where you are restoring the data.

Restoring Read-only Backups for a Coordinator [page 50]
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.

Parent topic: Restoring Database Backups [page 33]

Related Information

Database Restore [page 34]
Verifying Database Backups [page 37]
Displaying Header Information [page 39]
Restoring to Raw Devices [page 40]
Restoring Cache Dbspaces [page 42]
Moving Database Files [page 43]
Error Recovery [page 51]

3.7.1 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.

  > **i 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.
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*.

Procedure

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=<password>;
   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.

   > ⚠ Caution
   >
   > 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`.

Results

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.

Task overview: Restoring Multiplex Stores [page 45]

Related Information

Restoring Multiplex Stores to the Same Location [page 48]
Restoring Read-only Backups for a Coordinator [page 50]
Restoring Cache Dbspaces [page 42]

3.7.2 Restoring Multiplex Stores to the Same Location

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

Procedure

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).

   > **i** 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.
Windows	In 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 *SAP IQ 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=<password>;
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 *SAP IQ SQL Reference*.
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 *SAP IQ 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.

Task overview: Restoring Multiplex Stores [page 45]

Related Information

Restoring Multiplex Stores to a Different Location [page 46]
Restoring Read-only Backups for a Coordinator [page 50]
Back Up the Correct Files [page 23]
Restoring Cache Dbspaces [page 42]

3.7.3 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.

Context

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

Procedure

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 Steps

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).

Task overview: Restoring Multiplex Stores [page 45]

Related Information

Restoring Multiplex Stores to a Different Location [page 46]
Restoring Multiplex Stores to the Same Location [page 48]

3.8 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 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=<password>;dbf=node1/users/localhost/mydb.db;
links=tcpip{host=serv1;port=1234};eng=serv1_iqdemo'
```

Returns:

```
CONNECT DATABASE mydb USER DBA IDENTIFIED BY <password>
```

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.

Parent topic: Restoring Database Backups [page 33]

Related Information

Database Restore [page 34]
Verifying Database Backups [page 37]
Displaying Header Information [page 39]
Restoring to Raw Devices [page 40]
Restoring Cache Dbspaces [page 42]
Moving Database Files [page 43]
Restoring Multiplex Stores [page 45]

4 Point-in-Time Recovery Operations

Point-in-time recovery operations restore the database to a particular point in time.

In this section:

Point-in-Time Recovery [page 52]
: Point-in-time recovery restores all committed database operations prior to a specified point in time. A point in time can be expressed as a timestamp and timezone, or as a transaction log offset.

Alternate Database Timelines [page 59]
: A point-in-time recovery to a time earlier than the end-point of the point-in-time recovery logs creates an alternate database timeline. Subsequent point-in-time recovery operations may be done on any existing timeline

Retrieving Timeline Values [page 62]
: Methods that return details about current and previous timelines, logs, and log values.

Related Information

Database Backup [page 6]

4.1 Point-in-Time Recovery

Point-in-time recovery restores all committed database operations prior to a specified point in time. A point in time can be expressed as a timestamp and timezone, or as a transaction log offset.

Point-in-time recovery extends, but does not replace, standard backup and recovery operations. Point-in-time recovery always begins by restoring a database from a data backup, then replays all committed activity from the recovery logs that cover the entire time period between the backup and the specified point in time. Transactions that were in progress, but not committed, are rolled back.

Enabling Point-in-Time Recovery

1. Enable IQ_POINT_IN_TIME_RECOVERY_LOGGING:

   ```
   SET OPTION PUBLIC.IQ_POINT_IN_TIME_RECOVERY_LOGGING = 'ON'
   ```

2. Use ALTER DBSPACE to identify the directory where you want to archive the recovery logs:

   ```
   ALTER DBSPACE IQ_SYSTEM_LOG RENAME <new-directory-specification>/<file-prefix>
   ```

SAP IQ saves the point-in-time recovery log in the same directory as the `.db` file by default. This command saves the log in another directory. The `new-directory-specification` must point to an existing directory. On multiplex servers, the IQ_SYSTEM_LOG directory must reside on a shared file system and be writable by all multiplex nodes. For other constraints, see *Redirecting Log Output*.

3. Perform a data backup:

```
BACKUP DATABASE ...
    TO <archive_device>
```

Point-in-time recovery logging only becomes fully enabled when the data backup begins.

Setting the IQ_POINT_IN_TIME_RECOVERY_LOGGING option to OFF, disables point-in-time recovery immediately. To re-enable point-in-time recovery, you must complete all steps in this procedure, including a FULL, INCREMENTAL, or INCREMENTAL SINCE FULL backup.

Redirecting Log Output

By default, SAP IQ saves the point-in-time recovery log in the same directory as the `.db` file. To save the log in another directory, use:

```
ALTER DBSPACE IQ_SYSTEM_LOG RENAME <new-directory-specification>/file-prefix
```

Point `<new-directory-specification>` to an existing directory. If point-in-time recovery logging is active, using this command with a `<new-directory-specification>` opens a new log at the new location. For multiplex servers, this directory must reside on a shared file system and be writable by all multiplex nodes.

ALTER DBSPACE assumes that any string terminated by a slash (Windows) or backslash (UNIX) in the `<new-directory-specification>` is a directory name. A string not terminated by a slash or backslash is prefixed to the log name.

Log Size

The IQ_LOG_MAX_SIZE option limits the size of the log:

```
SET OPTION PUBLIC.IQ_LOG_MAX_SIZE = '<megabytes>'
```

The default setting of 0 allows the log size to increase without limit. If you limit the log size, and the file exceeds the limit, the server closes the current log, and creates a new log in the same directory. Subsequent log entries are written to the new log.

Log Names

All recovery logs include a `.iqlog` extension. Token position within the file name provide statistics about the log:

Position	Description
<prefix>	A <prefix> is optional. If the ALTER DBSPACE `new-directory-specification` includes a terminating string without a slash or backslash, the terminating string is prefixed to the log name.
_<nnn>	The first 3 digits in the log name is the server ID. SAP IQ always assigns `_000` as the server ID to simplex servers.
_<nnnnnn>	The 6 digits that follow the server identifier, is the log sequence number. If IQ_POINT_IN_TIME_RECOVERY_LOGGING is ON, each log backup or `ALTER DBSPACE IQ_SYSTEM_LOG RENAME` closes the current log and opens a new one. Each new log increments the log sequence number.
_<nnnn>	The 4 digits that follow the log sequence number is the log sub-sequence number. A new log that opens after a backup or `ALTER DBSPACE IQ_SYSTEM_LOG RENAME` starts with a sub-sequence equal to 1. If the archive size exceeds the value set by IQ_LOG_MAX_SIZE, the server closes the current log and opens a new one. The sub-sequence number is incremented to reflect the change.

Automatic Backups

To back up the logs required by point-in-time recovery at set time intervals, use:

```
SET OPTION PUBLIC.IQ_LOG_BACKUP_INTERVAL= '<minutes>'
```

Time intervals are expressed in minutes. The IQ_LOG_BACKUP_INTERVAL option backs up the transaction log and RLV log. The point-in-time recovery logs (`.iqlog`) are not included in the backup because of size considerations. The IQ_LOG_BACKUP_INTERVAL option must be explicitly set. The minimum supported backup interval is 5 minutes. Default behavior requires manual backups of the activity logs.

Manual Backups

To backup the point-in-time recovery logs manually, use:

```
BACKUP DATABASE ...
    POINT IN TIME RECOVERY LOGS ONLY TO '<target-directory>/<archive-file-name>'
```

SAP IQ closes the current log and starts a new one. Copies of the previous logs are left behind as backups. The archive is written to the `target-directory`.

POINT IN TIME RECOVERY LOGS ONLY supports only one TO clause, which points to a location on the file system. Name of the archive is user-determined. No other options are allowed. The POINT IN TIME RECOVERY

LOGS ONLY clause can only be used to back up activity logs for point-in-time recovery. Do not use this clause as part of a data backup.

RESTORE and Recovery

Always begin a point-in-time recovery by restoring a database from a full data backup, then apply all incremental backups prior to the point in time you want to recover. Use the RESTORE DATABASE point-in-time recovery syntax extensions as a final step to replay all committed activity from the recovery logs that cover the entire time period between the backup and the period just prior to the point in time you specify:

```
RESTORE DATABASE '<db_file>'
   [ FROM '< file-name >' <...>
   [ USING LOG PATH  ' directory, [...]]
RECOVER UNTIL [ TIMESTAMP <timestamptz> | OFFSET <log-offset> ]
  OVERWRITE EXISTING
  CLEAR LOG
  ON TIMELINE <GUID>
```

Always specify a timestamp that is greater than the backup time of the data backup specified in the FROM clause of the restore command. This ensures that the database includes all committed transactions in the recovery logs. If the specified point in time is earlier than the last checkpoint in the backup database, the server returns an error.

OVERWRITE EXISTING

Point-in-time recovery, unless explicitly specified, restores the database in the same location. Point-in-time recovery fails during restore if it finds the existing database in the same location where it attempts to restore. During point-in-time recovery to the same location; supply an OVERWRITE EXISTING clause that tells SAP IQ to overwrite existing dbfiles and transaction logs during a RESTORE. The OVERWRITE EXISTING clause in this instance eliminates the need to drop the existing database manually.

CLEAR LOG

During point-in-time recovery, the restore automatically backs up the existing transaction log and RLV log. Every point-in-time restore looks for the existing transaction log, RLV log, and point in time logs in the existing database directories. The restore operation returns an error if it fails to locate these logs. If you want to restore to a new environment, however, and have no existing logs to back up, supply a RESTORE command with the CLEAR LOG clause to stop SAP IQ from seeking existing log files in the database directories.

ON TIMELINE

You can optionally specify the ON TIMELINE clause to specify the timeline to use for point-in-time recovery.

! Restriction

- SAP IQ does not support RENAME with point-in-time recovery.
- SAP IQ does not support point-in-time recovery on multiplex systems.
- The CLEAR LOG clause cancels the automatic log backup. An automatic backup of the restored database still occurs.

Performing Point-in-Time Recovery

To perform point-in-time recovery:

1. Shut down the database server.
2. Start a server that you can use to connect to the utility database:

   ```
   start_iq -su <password> -gd DBA -gm 1 -n my_server
   ```

3. Start dbisql and connect to the utility database:

   ```
   dbisql -c "UID=DBA;PWD=<password>;DBN=utility_db"
   ```

4. Restore the database prior to the point-in-time you want to recover.
   Start with a full database backup. Repeat this step for every incremental backup prior to the point in time you want to recover.
5. Run `RESTORE DATABASE` with the point-in-time syntax extensions.
   Perform the recovery with appropriate incremental backup. See the Recovery Scenarios for syntax examples.
6. Shut down the server and utility database.
7. Start the database server normally.

Recovery Scenarios

> **i** Note
>
> If you use `isql` to run the following scenarios, CHAINED is set to OFF and each statement commits immediately. If you use `dbisql`, however, CHAINED is set to ON by default, so that the data you insert remains uncommitted when the backup runs, preventing the data from being captured. To avoid this when using `dbisql`, either add COMMIT before you run each backup, or add "SET TEMPORARY OPTION CHAINED=OFF;" at the beginning of each script.

Scenario 1

Point-in-time recovery using point-in-time recovery logs only:

```
// enable Point in time recovery
SET OPTION PUBLIC.IQ_POINT_IN_TIME_RECOVERY_LOGGING = 'ON';
ALTER DBSPACE IQ_SYSTEM_LOG RENAME '/demo/pitrLog/PITRLOG';
BACKUP DATABASE FULL to '/demo/dataBackup/FULL1';
// perform some DDL/DML operations
CREATE TABLE T1 ( ID INT );
INSERT INTO T1 VALUES (1);
// record timestamp TS1 / OFFSET OFS1
TS1: select getdate()
OFS1: select db_property('LastCommitRedoPos')
// perform DDL/ DML operations
CREATE TABLE T2 ( ID INT );
INSERT INTO T2 VALUES (1);
// record timestamp TS2 / OFFSET OFS2
TS2: select getdate()
OFS2: select db_property('LastCommitRedoPos')
// restore the database to point in time TS1 / OFS1
// shut down the database
```

```
// start a utility server
RESTORE DATABASE 'iqdemo' FROM '/demo/dataBackup/FULL1'
    RECOVER UNTIL TIMESTAMP '<TS1>'
    USING LOG PATH '/demo/pitrLog/' ,
    OVERWRITE EXISTING;
// or
RESTORE DATABASE 'iqdemo' FROM '/demo/dataBackup/FULL1'
    RECOVER UNTIL OFFSET <OFS1>
    USING LOG PATH '/demo/pitrLog/'
    OVERWRITE EXISTING;
```

Scenario 2

Point-in-time recovery using point-in-time recovery logs and point-in-time recovery log backup archives:

```
// enable point-in-time recovery
SET OPTION PUBLIC.IQ_POINT_IN_TIME_RECOVERY_LOGGING = 'ON';
ALTER DBSPACE IQ_SYSTEM_LOG RENAME '/demo/pitrLog/PITRLOG';
BACKUP DATABASE FULL to '/demo/dataBackup/FULL1';
// perform DDL / DML operations
CREATE TABLE T1 ( ID INT );
INSERT INTO T1 VALUES (1);
// record timestamp TS1 / OFFSET OFS1
TS1: select getdate()
OFS1: select db_property('LastCommitRedoPos')
// perform DDL / DML operations
CREATE TABLE T2 ( ID INT );
INSERT INTO T2 VALUES (1);
// record timestamp TS2 / OFFSET OFS2
TS2: select getdate()
OFS2: select db_property('LastCommitRedoPos')
// perform a PITR log backup
BACKUP DATABASE
    POINT IN TIME RECOVERY LOGS ONLY TO '/demo/pitrLogBackup/PITR1';
// perform DDL/ DML operations
CREATE TABLE T3 ( ID INT );
INSERT INTO T3 VALUES (1);
// restore the database to point-in-time TS2 OR OFS2
// shut down the database
// start a utility server
RESTORE DATABASE 'iqdemo'
    FROM '/demo/dataBackup/FULL1'
    RECOVER UNTIL TIMESTAMP '<TS2>'
    USING LOG PATH '/demo/pitrLogBackup/', '/demo/pitrLog/'
    OVERWRITE EXISTING;
// or
RESTORE DATABASE 'iqdemo'
    FROM '/demo/dataBackup/FULL1'
    RECOVER UNTIL OFFSET <OFS2>
    USING LOG PATH '/demo/pitrLogBackup/', '/demo/pitrLog/'
    OVERWRITE EXISTING;
```

Scenario 3

Point-in-time recovery on an RLV-enabled database with an incremental backup:

```
// perform a full database backup
BACKUP DATABASE FULL TO '/demo/dataBackup/FULL1';
// perform DDL / DML operations
CREATE TABLE T0 ( ID INT );
INSERT INTO T0 VALUES (1);
RECORD TIME STAMP TS0 / OFFSET OFS0;
// enable point-in-time recovery using an incremental backup
SET OPTION PUBLIC.IQ_POINT_IN_TIME_RECOVERY_LOGGING = 'ON';
ALTER DBSPACE IQ_SYSTEM_LOG RENAME '/demo/pitrLog/PITRLOG';
BACKUP DATABASE INCREMENTAL to '/demo/dataBackup/INCR1';
```

```
// perform DDL / DML operations
CREATE TABLE T1 ( ID INT );
INSERT INTO T1 VALUES (1);
// record timestamp TS1 / OFFSET OFS1
TS1: select getdate()
OFS1: select db_property('LastCommitRedoPos')
// perform DDL / DML operations
CREATE TABLE T2 ( ID INT );
INSERT INTO T2 VALUES (1);
// record timestamp TS2 / OFFSET OFS2
// perform a PITR log backup
BACKUP DATABASE
    POINT IN TIME RECOVERY LOGS ONLY TO '/demo/pitrLogBackup/PITR1';
// note the RLV DBspace files and their location
select * from sp_iqfile() where segmentType = 'RLV'";
// restore the database to point-in-time TS1 OR OFS1
// shut down the database
// Do a physical file backup of all files in the RLV Dbspace.
// start a utility server
// Perform a restore of database from backup FULL1
RESTORE DATABASE 'iqdemo'
    FROM '/demo/dataBackup/FULL1'
    OVERWRITE EXISTING;
// restore the RLV Dbspace files from the backup taken above to its original
location
// perform a point-in-time recovery to time TS1 using incremental backup INCR1
RESTORE DATABASE 'iqdemo'
    FROM '/demo/dataBackup/INCR1'
    RECOVER UNTIL TIMESTAMP '<TS1>'
    USING LOG PATH '/demo/pitrLogBackup/PITR1/'
    OVERWRITE EXISTING CLEAR LOG;
```

Backup History

Use the `sp_iqpurgebackuphistory` stored procedure to manage backup history. `sp_iqpurgebackuphistory` deletes all rows from the `SysIQBackupHistory` and `SysIQBackupHistoryDetail` system tables that match criteria defined by four parameters:

Parameter	Type	Description
`bu_id`	UNSIGNED BIGINT	Deletes entries that match the `bu_id`: `sp_purgeiqbackuphistory(bu_id ='9277')`
`bu_time_low`	TIMESTAMP	Deletes entries with timestamps (hh:mm:ss.ms) greater than or equal to `bu_time_low`: `sp_purgeiqbackuphistory( bu_time_low='2013/01/01')`
`bu_time_high`	TIMESTAMP	Deletes entries with backup times less than or equal to `bu_time_high`: `sp_purgeiqbackuphistory( bu_time_high='2013/01/01' )`

Parameter	Type	Description
bu_type	TINYINT	Deletes entries that match the bu_type: • 0 = FULL • 1 = INCREMENTAL • 2= INCREMENTAL SINCE FULL

Since SysIQBackupHistory and SysIQBackupHistoryDetail are system tables, purging entries is non-transactional and cannot be rolled back.

Additional Information

- See the IQ_POINT_IN_TIME_RECOVERY_LOGGING, IQ_POINT_IN_TIME_RECOVERY_LOG_INTERVAL, IQ_LOG_INTERVAL database options in *SAP IQ SQL Reference*
- See ALTER DBSPACE, BACKUP DATABASE, and RESTORE DATABASE statements in*SAP IQ SQL Reference*
- See sp_purgeiqbackuphistory procedure in *SAP IQ SQL Reference*

Parent topic: Point-in-Time Recovery Operations [page 52]

Related Information

Alternate Database Timelines [page 59]
Retrieving Timeline Values [page 62]

4.2 Alternate Database Timelines

A point-in-time recovery to a time earlier than the end-point of the point-in-time recovery logs creates an alternate database timeline. Subsequent point-in-time recovery operations may be done on any existing timeline

Alternate timelines are identified by globally unique identifiers (GUID) that establish the current timeline, previous timeline, current log sequence, and previous log sequence. The previous timeline GUID includes an OFFSET reference to the position within the previous timeline from which the current timeline branched. All GUIDs carry a timestamp created at the time the point-in-time recovery operation begins.

Creating Alternate Timelines

To create an alternate timeline, run RESTORE DATABASE with the appropriate point-in-time recovery syntax extensions to recover to a specific <timestampz> or <log-offset>:

```
RESTORE DATABASE ' <db_file> '
          FROM ' <file-name> '
          RECOVER UNTIL TIMESTAMP ' <timestamptz> '
          USING LOG PATH ' <directory 1> ,  <directory 2>, ... '
```

or

```
RESTORE DATABASE ' <db_file> '
          FROM ' <file-name> '
          RECOVER UNTIL OFFSET <log-offset>
          USING LOG PATH ' <directory 1> ,  <directory 2>, ... '
```

Point-in-time recovery resumes logging operations along the new timeline when you restart the database server.

Initial and Alternate Timelines

Point-in-time recovery can support any number of timelines. An initial database timeline is established by the data backup you perform to enable point-in-time recovery. A point-in-time recovery to any time earlier than the end-point of current timeline creates an alternate database timeline.

Consider these timelines:

```
 DB ---- LB1 --  -- LB2 ---- P1 ---- LB3 ---- T1
                |
              PITR1 ---- LB4 --  -- LB5 ---- P2 ---- T2
                                 |
                              PITR2 ---- LB6 ---- P3 ---- LB7 ---- T3
```

DB is the data backup and LB is a log backup. T1 represents the default timeline. PITR1 represents a recovery operation performed to a point between LB1 and LB2. This started timeline T2. PITR2 represents a recovery operation performed to a point between LB4 and LB5. This started timeline T3.

Restoring to an Alternate Timeline

All point-in-time recovery operations that restore to an alternate timeline include an ON TIMELINE clause:

```
RESTORE DATABASE ' db_file '
 [ FROM ' file-name '
  USING LOG PATH ' directory, [...]]
  RECOVER UNTIL  [ TIMESTAMP  <timestamptz> | OFFEST <log-offset> ]
  ON TIMELINE ' <GUID> '
  OVERWRITE EXISTING
  CLEAR LOG
```

The ON TIMELINE clause requires a timeline GUID that identifies the alternate timeline. Point-in-time recovery operations without this clause restore to the current timeline. Use the OVERWRITE EXISTING clause to overwrite existing dbfiles and transaction logs.

If RESTORE DATABASE cannot locate the transaction log, RLV log, and point-in-time recovery logs during a point-in-time recovery operation, the recovery operation fails. This can happen in cases when the database is restored to a new location and there are no logs available in the new environment. In cases like this, use the CLEAR LOG clause to cancel the automatic log backups that normally occur during a point-in-time recovery operation.

CLEAR LOG cancels the automatic log backup. An automatic backup of the restored database still occurs.

Recovery Scenarios

Use a backup on the initial timeline (T0) to create a secondary timeline (T1).

```
//record the current timeline (T0) GUID
SELECT db_property('CurrentTimelineID')
//  enable point in time recovery
SET OPTION PUBLIC.IQ_POINT_IN_TIME_RECOVERY_LOGGING = 'ON'
ALTER DBSPACE IQ_SYSTEM_LOG RENAME '/localhost/demo/pitrLog/PITRLOG'
BACKUP DATABASE FULL TO '/localhost/demo/dataBackup/FULL1'
// perform some DDL/DML operations
CREATE TABLE T1 ( ID INT )
INSERT INTO T1 VALUES (1)
// record timestamp TS1 / offset OFS1
TS1: SELECT getdate()
OFS1: SELECT db_property('LastCommitRedoPos')
//  perform DDL/ DML operations
CREATE TABLE T2 ( ID INT )
INSERT INTO T2 VALUES (1)
//  record timestamp TS2 / offset OFS2
TS2: select getdate()
OFS2: select db_property('LastCommitRedoPos')
// shutdown the database
// start a utility server
// connect to the utility server
// recover to offset OFS2 in timeline TS2
RESTORE DATABASE 'iqdemo' FROM '/localhost/demo/dataBackup/FULL1'
    RECOVER UNTIL OFFSET OFS2
    USING LOG PATH '/localhost/demo/pitrLog/'
    OVERWRITE EXISTING
// record the new timeline (T1) GUID - TLID1
SELECT db_property('CurrentTimelineID')
```

Use a backup on the initial timeline (T0) to create a third timeline (T2). The current timeline is timeline T1.

```
//  perform some DDL/DML operations
CREATE TABLE T3 ( ID INT )
INSERT INTO T3 VALUES (1)
// record timestamp TS3 / offset OFS3
TS1: SELECT getdate()
OFS1: SELECT db_property('LastCommitRedoPos')
// perform DDL/ DML operations
CREATE TABLE T4 ( ID INT )
INSERT INTO T4 VALUES (1)
// record timestamp TS4 / offset OFS4
TS2: SELECT getdate()
OFS2: SELECT db_property('LastCommitRedoPos')
// shutdown the database
```

```
// start a utility serveR
// connect to the utility server
// restore to timestamp TS3 / offset OFS3
RESTORE DATABASE 'iqdemo'
    FROM '/localhost/demo/dataBackup/FULL1'
    RECOVER UNTIL OFFSET OFS3
    USING LOG PATH '/localhost/demo/pitrLog/'
    OVERWRITE EXISTING
// record the new timeline (T2) GUID TLID2
SELECT db_property('CurrentTimelineID')
```

Return to a timestamp on an older timeline. Timeline (T3) is a new timeline on timeline (T0). Assume that the current timline is T2.

```
// perform some DDL/DML operations
CREATE TABLE T5 ( ID INT )
INSERT INTO T5 VALUES (1)
// record timestamp TS5 / offset OFS5
TS1: select getdate()
OFS1: select db_property('LastCommitRedoPos')
// shutdown the database
// start a utility server
// connect to the utility server and execute the restore command
// recover to timestamp TS1 / offset OFS1 on timeline T0
RESTORE DATABASE 'iqdemo' FROM '/localhost/demo/dataBackup/FULL1'
    RECOVER UNTIL OFFSET <OFS1>
    USING LOG PATH '/localhost/demo/pitrLog/'
    OVERWRITE EXISTING ON TIMELINE '<TLID1>'
// record the new timeline (T3) GUID TLID3
SELECT db_property('CurrentTimelineID')
```

Parent topic: Point-in-Time Recovery Operations [page 52]

Related Information

Point-in-Time Recovery [page 52]
Retrieving Timeline Values [page 62]

4.3 Retrieving Timeline Values

Methods that return details about current and previous timelines, logs, and log values.

db_backupheader Utility

Reads the first backup archive, returns backup statistics, timeline and log details:

```
db_backupheader [ path ] backup_file
```

db_backupheader is a command line utility. See *db_backupheader Utility* in *SAP IQ Utility Guide* for additional information.

Timeline Properties

Start dbisq, connect to the database, and select these database properties to retrieve information about current and previous timelines.

Property Name	Description
CurrentRedoPos	Current log redo position: `select db_property('CurrentRedoPos')` Returns a numeric OFFSET value. Results are similar to: '1573909'.
CurrentTimelineID	Globally unique identifier for the current database timeline: `select db_property('CurrentTimelineID')` Returns a 36 character string. Results are similar to: '670e7768-34ae-11e4-8000-a6da5f46c1e9'.
CurrentTimelineUTCCreationTime	UTC timestamp which identifies when the current database timeline was created: `select db_property('CurrentTimelineUTCCreationTime')` Returns a UTC timestamp. Results are similar to: '2014-09-05 11:40:38.591+00:00'.
CurrentTransactionLogID	Globally unique identifier for the current transaction log: `select db_property('CurrentTransactionLogID')` Returns a 36 character string. Results are similar to: '6a0c9c38-34ae-11e4-8000-a6da5f46c1e9'.
IQPointInTimeRecoveryLogFilename	Current point-in-time recovery log name and location: `select db_property('IQPointInTimeRecoveryLogFilename')` Returns the path to the point-in-time recovery log. Results are similar to: 'demo/pitrLog/PITR__000_000001_0001.iqlog'.
IQPointInTimeRecoveryLogStatus	Current status of point-in-time recovery log: `select db_property('IQPointInTimeRecoveryLogStatus')` Returns an ON, OFF, or Requested value. Requested is returned when the IQ_POINT_IN_TIME_RECOVERY_LOGGING option is set, but actual logging has not been turned ON.

Property Name	Description
LastCheckpointTime	Last checkpoint time: `select db_property('LastCheckpointTime')` Returns a UTC timestamp. Results are similar to: '2014-09-09 11:28:04.219'.
LastCommitRedoPos	Redo position at end of last commit: `select db_property('LastCommitRedoPos')` Returns a numeric OFFSET value. Results are similar to: '1574242'.
LastSyncedRedoPos	Last redo position written and synchronized to disk: `select db_property('LastSyncedRedoPos')` Returns a numeric OFFSET value. Results are similar to: '1574269'.
LastWrittenRedoPos	Last redo position written to disk: `select db_property('LastWrittenRedoPos')` Returns a numeric OFFSET value. Results are similar to: '1574296'.
PreviousTimelineID	Globally unique identifier for the previous database timeline: `select db_property('PreviousTimelineID')` Returns a 36 character string. Results are similar to: '98b69a5c-2c0c-11e4-8000-853d9a9151da'.
PreviousTimelineUTCCreationTime	UTC timestamp that identifies when the previous database timeline was created: `select db_property('PreviousTimelineUTCCreationTime')` Returns a 36 character string. Results are similar to: '2014-08-25 12:02:14.128566+00:00'.
PreviousTransactionLogID	Globally unique identifier for the previous transaction log: `select db_property('PreviousTransactionLogID')` Returns a 36 character string. Results are similar to: '9bf0b6bc-2c0c-11e4-8000-853d9a9151da'.
TimelineBranchOffset	Transaction log offset in the previous database timeline from which the current timeline was branched: `select db_property('TimelineBranchOffset')` Returns a numeric OFFSET value. Results are similar to: '1341357'.

Parent topic: Point-in-Time Recovery Operations [page 52]

Related Information

Point-in-Time Recovery [page 52]
Alternate Database Timelines [page 59]

5 Shared-Nothing Multiplex Backup and Recovery

In a shared-nothing multiplex where primary data is stored in DAS dbfiles, the coordinator archives SAN dbspaces, whereas all primary DAS dbfiles in DAS dbspaces are archived locally on each secondary node.

For detailed information on configuring and using shared-nothing multiplex architecture for massively parallel processing, see *SAP IQ Administration: Multiplex*.

> **i** Note
>
> Shared-nothing multiplex does not support point in time recovery in this release of SAP IQ.

In this section:

Shared-Nothing Multiplex Full Backup [page 67]
: An example backup procedure where you perform a FULL backup on a multiplex containing primary and mirror DAS dbfiles.

Shared-Nothing Multiplex Full Restore [page 67]
: An example restore procedure restoring a multiplex with primary and mirror DAS dbfiles without changing node paths or parameters.

Shared-Nothing Multiplex Incremental Backup [page 68]
: An example backup procedure where you perform a FULL backup, change the shared-nothing multiplex configuration, and then perform an INCREMENTAL backup.

Shared-Nothing Multiplex Incremental Restore with Configuration Changes [page 69]
: An example restore procedure restoring a multiplex with primary and mirror DAS dbfiles without changing node paths or parameters. The multiplex configuration changed between the FULL and INCREMENTAL backups.

Shared-Nothing Multiplex Incremental Restore to a Different Location [page 70]
: In a shared-nothing multiplex, you use the ALTER DATABASE statement to change node paths or parameters, instead of dropping and re-adding changed nodes. This example restores primary and mirror DAS dbfiles while changing coordinator and secondary server node paths and parameters. The multiplex configuration changed between the FULL and INCREMENTAL backups.

5.1 Shared-Nothing Multiplex Full Backup

An example backup procedure where you perform a FULL backup on a multiplex containing primary and mirror DAS dbfiles.

Procedure

1. Create a multiplex with this configuration:
   - {Coordinator, Host0}
   - {Server1, Host1, DAS1, DAS2}
   - {Server2, Host2, DAS3, Mirror1}
   - {Server3, Host3, Mirror2, Mirror3}
2. Perform a FULL data backup.

   ```
   BACKUP DATABASE FULL TO '/home/save/full_bkup'
   ```

   Unlike a shared dbspace, where the coordinator creates a single SAN archive with a name like `'full_bkup.1'`, backup creates different archives for hosts containing primary DAS dbfiles. The server ID is suffixed to the archive filename. In this example:
   - The archive created for 'Server1' is `'full_bkup.<Server1_ID>'`.
   - The archive created for 'Server2' is `'full_bkup.<Server2-ID>.1'`. Server 2 will write a DAS archive for dbfiles DAS3: `'full_bkup.<server3-id>.1'`
   - Server 3 contains only mirrors and will not write a DAS archive. However, this node needs to be up and running during database restore.

5.2 Shared-Nothing Multiplex Full Restore

An example restore procedure restoring a multiplex with primary and mirror DAS dbfiles without changing node paths or parameters.

Context

This example restores from full backup archive 'full_bkup'.

Procedure

1. Start the utility database from Host0 (the same location as the coordinator).
2. Restore the SAN from 'full_bkup' on the coordinator. Do not use the `MULTIPLEX` keyword.

   ```
   RESTORE DATABASE 'mydatabase' FROM '/home/save/full_bkup'
   ```
3. Start the database as coordinator.
4. Synchronize and start all secondary nodes from their respective locations.
5. Restore the DAS dbspace data from 'full_bkup'.
   Call `RESTORE DATABASE` with the `MULTIPLEX` keyword.

   ```
   RESTORE DATABASE 'mydatabase' MULTIPLEX FROM '/home/save/full_bkup'
   ```
6. Change all DAS dbspaces to online.

5.3 Shared-Nothing Multiplex Incremental Backup

An example backup procedure where you perform a FULL backup, change the shared-nothing multiplex configuration, and then perform an INCREMENTAL backup.

Context

This example assumes you are making a full backup archive 'full_bkup', and an incremental backup archive 'inc_bkup'.

The initial multiplex configuration is:

- {Coordinator, Host0}
- {Server1, Host1, DAS1, DAS2}
- {Server2, Host2, DAS3, Mirror1}
- {Server3, Host3, Mirror2, Mirror3}

Procedure

1. Perform a FULL data backup.

   ```
   BACKUP DATABASE FULL TO '/home/save/full_bkup'
   ```
2. Change the shared-nothing multiplex configuration. Drop existing Server2 and add a new secondary node:

   {Server4, Host4, DAS4}

3. Perform an incremental backup.

```
BACKUP DATABASE INCREMENTAL TO '/home/save/inc_bkup'
```

   Backup creates different archives for hosts containing primary DAS dbfiles. The server ID is suffixed to the archive filename. In this example, the archive created for 'Server1' is `'full_bkup.<Server1_ID>'`.

5.4 Shared-Nothing Multiplex Incremental Restore with Configuration Changes

An example restore procedure restoring a multiplex with primary and mirror DAS dbfiles without changing node paths or parameters. The multiplex configuration changed between the FULL and INCREMENTAL backups.

Prerequisites

You have completed a shared-nothing multiplex incremental backup.

Context

This example assumes you have a full backup archive 'full_bkup', and an incremental backup archive 'inc_bkup'.

This example assumes you have dropped secondary node server2 and added secondary node server3 in between the two backups.

Procedure

1. Start the utility database from Host0 (the same location as the coordinator).
2. Restore the SAN from 'full_bkup' on the coordinator. Do not use the `MULTIPLEX` keyword.

```
RESTORE DATABASE 'mydatabase' FROM '/home/save/full_bkup'
```

3. Start the database you restored in the previous step. Start it on the coordinator node.
4. Run `sp_iqmpxinfo` and note secondary node information including:
   - server_id
   - server_name
   - connection_info
   - db_path

5. Synchronize and start all secondary nodes in the multiplex configuration (server2 in this example). Start them from their respective locations.
6. Restore the DAS dbspace data from 'full_bkup'.

   Call `RESTORE DATABASE` with the `MULTIPLEX` keyword.

   ```
   RESTORE DATABASE 'mydatabase' MULTIPLEX FROM '/home/save/full_bkup'
   ```
7. Shutdown all servers.
8. Start the utility database from Host0 (the same location as the coordinator).
9. Restore the SAN from the incremental backup 'inc_bkup'. Do not use the `MULTIPLEX` keyword.

   ```
   RESTORE DATABASE 'mydatabase' FROM '/home/save/inc_bkup'
   ```
10. Start the database as coordinator.
11. Run `sp_iqmpxinfo` and note secondary node information including:
    - server_id
    - server_name
    - connection_info
    - db_path

    Notice that server2 has been dropped and is no longer part of the multiplex configuration. You will not start server2.

    Notice that server4 is a new server in the multiplex, and has a DAS dbfile. You will start server3.
12. Synchronize and start all secondary nodes in the multiplex configuration (server3 in this example) from their respective locations. You noted these secondary nodes in the previous step.
13. Restore the DAS dbspace from the incremental archive.

    Call `RESTORE DATABASE` with the `MULTIPLEX` keyword for each node in the logical server with a primary or mirror DAS dbfile. Restore each node from 'inc_bkup'. Avoid starting databases between incremental restores.

    ```
    RESTORE DATABASE 'mydatabase' MULTIPLEX FROM '/home/save/inc_bkup'
    ```
14. Change all DAS dbspaces to online.

5.5 Shared-Nothing Multiplex Incremental Restore to a Different Location

In a shared-nothing multiplex, you use the ALTER DATABASE statement to change node paths or parameters, instead of dropping and re-adding changed nodes. This example restores primary and mirror DAS dbfiles while changing coordinator and secondary server node paths and parameters. The multiplex configuration changed between the FULL and INCREMENTAL backups.

Prerequisites

You have completed a shared-nothing multiplex incremental backup.

Context

This example assumes you have a full backup archive 'full_bkup', and an incremental backup archive 'inc_bkup'.

This example assumes you are restoring the coordinator to:

- A new host 'machinex' (10.9.248.149) and port 5341
- A new location '/machinex8/users/myname/main_nbld_solAMD_64p/mpxstore/mpx_configdb.db'

This example drops secondary node server2 and adds secondary node server4.

For each step that includes a RESTORE command, specify a RENAME clause to restore certain dbspace files to a different path.

Procedure

1. Start the utility database on the new host at the new coordinator location.
2. Restore the SAN from 'full_bkup' on the coordinator. Do not use the `MULTIPLEX` keyword.

   ```
   RESTORE DATABASE 'mydatabase' FROM '/home/save/full_bkup'
   ```
3. Start the database with override switch (-iqmpx_ov 1).
4. Change the database location, host, and port for the coordinator using the `ALTER MULTIPLEX SERVER` command.

   ```
   ALTER MULTIPLEX SERVER dbsrv612_nc15341
   DATABASE '/machinex8/users/myname/main_nbld_solAMD_64p/mpxstore/
   mpx_configdb.db'
   HOST '10.9.248.149' PORT 5341
   ```
5. Change any parameters (host and port, for example) for secondary nodes.

   ```
   ALTER MULTIPLEX SERVER dbsrv612_nw49837
   DATABASE '/machinex8/users/myname/main_nbld_solAMD_64p/w/mpx_configdb.db'
   HOST '10.9.248.149' PORT 9837
   ```
6. Start the database, as coordinator, from the new location.
7. Run `sp_iqfile` and note which servers have DAS dbfiles. You will need to start these servers in a later step.
8. Run `sp_iqmpxinfo` and note secondary node information including:
   - server_id
   - server_name
   - connection_info

- db_path

9. Synchronize the secondary nodes. Start them from their respective locations.
10. Restore the DAS dbspace data from 'full_bkup'.

    Call `RESTORE DATABASE` with the `MULTIPLEX` keyword.

    ```
    RESTORE DATABASE 'mydatabase' MULTIPLEX FROM '/home/save/full_bkup'
    ```

11. Shut down all servers.
12. Start the utility database from Host0 (the same location as the coordinator).
13. Restore the SAN from the incremental backup 'inc_bkup'. Do not use the `MULTIPLEX` keyword.

    ```
    RESTORE DATABASE 'mydatabase' FROM '/home/save/inc_bkup'
    ```

14. Start the database as coordinator.
15. Run `sp_iqfile` and note which servers have DAS dbfiles. You will need to synchronize and start these servers in a later step.
16. Run `sp_iqmpxinfo` and note secondary node information including:
    - server_id
    - server_name
    - connection_info
    - db_path

    Notice that server2 has been dropped and is no longer part of the multiplex configuration. You will not start server2.

    Notice that server4 is a new server in the multiplex, and has a DAS dbfile. You will start server4.
17. Synchronize and start the secondary nodes from their respective locations. You noted these secondary nodes in the previous step.
18. Restore the DAS dbspace from the incremental archive.

    Call `RESTORE DATABASE` with the `MULTIPLEX` keyword for each node in the logical server with a primary or mirror DAS dbfile. Restore each node from 'inc_bkup'. Avoid starting databases between incremental restores.

    ```
    RESTORE DATABASE 'mydatabase' MULTIPLEX FROM '/home/save/inc_bkup'
    ```

19. Change all DAS dbspaces to online and read-write.

6 System Recovery and Database Repair

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

When you restart the database server, SAP 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.

In this section:

Recovery and Repair Overview [page 73]
: If your SAP IQ server or database has problems restarting, use this information to diagnose database startup problems, verify the consistency of databases, and repair databases.

Normal Recovery [page 74]
: During system recovery, any uncommitted transactions are rolled back, and any disk space used for old versions (that is, snapshots of database pages that were being used by transactions that did not commit) returns to the pool of available space.

Database Verification [page 74]
: Use `sp_iqcheckdb` for database verification.

Database Repair [page 82]
: Allocation problems can be repaired by running `sp_iqcheckdb` in `dropleaks` mode.

Forced Recovery Mode [page 89]
: Forced database recovery differs from normal database recovery.

Problems Reported by DBCC [page 93]
: Messages are reported for problems that DBCC cannot repair.

DBCC Error Messages [page 96]
: These are the most important messages in the DBCC output.

6.1 Recovery and Repair Overview

If your SAP 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 SAP Technical Support if necessary.

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

6.2 Normal Recovery

During system recovery, any uncommitted transactions are rolled back, and any disk space used for old versions (that is, 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 that can be accessed by secondary servers.

During recovery from a system failure or normal system shutdown, SAP 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 IQ server with at least as many connections as were actually in use when the server stopped.

6.3 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.

In this section:

Database Verification Using the sp_iqcheckdb Stored Procedure [page 75]
The SAP IQ Database Consistency Checker (DBCC) performs database verification. The `sp_iqcheckdb` stored procedure, in conjunction with server startup options, is the interface to DBCC.

Analyzing sp_iqcheckdb Output [page 78]
The output of `sp_iqcheckdb` consists of an extensive list of statistics and any errors reported by DBCC.

Resource Issues Running sp_iqcheckdb [page 81]
`sp_iqcheckdb` reports resource issues encountered while executing.

6.3.1 Database Verification Using the sp_iqcheckdb Stored Procedure

The SAP 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.

> **i** 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.

> **i** 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`

> **i** 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 HG table t2'`	Allocation checking of all HG 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 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'
```

> **i** Note
>
> Use `sp_iqrebuildindex` to repair index errors.

Related Information

sp_iqcheckdb Procedure [page 116]

6.3.2 Analyzing 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 (*****).

> **i** Note
>
> 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, where `<file_name>` is the name of the file to receive the output:

```
sp_iqcheckdb >#  <file_name>
```

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                 |*****
```

Block Ownership Conflicts

If you run `sp_iqcheckdb` in allocation mode and see the messages `** DBCC Status Errors Detected` and `** 1st Multiple Owner PBN`, a block ownership conflict was found.

> **i** Note
>
> Block ownership conflicts are only analyzed when the target of `sp_iqcheckdb` is either a database or a dbspace.

The following sample output indicates a block ownership conflict error:

```
============= ================================   ======
DBCC Allocation Mode Report
============== ================================   ======
** DBCC Status       Errors Detected                 ******

. . .
** 1st Multiple Owner PBN                411552      ******
** 1st Unowned PBN                       411543
. . .
** Blocks with Multiple Owners                       ******
** DBA.dupowned1.ASIQ_IDX_T1707_C1_FP    411552      ******
** DBA.dupowned2.ASIQ_IDX_T1708_C1_FP    411552      ******
** DBA.dupowned1.ASIQ_IDX_T1707_C1_FP    411553      ******
** DBA.dupowned2.ASIQ_IDX_T1708_C1_FP    411553      ******
** DBA.dupowned1.ASIQ_IDX_T1707_C1_FP    411554      ******
** DBA.dupowned2.ASIQ_IDX_T1708_C1_FP    411554      ******
** DBA.dupowned1.ASIQ_IDX_T1707_C1_FP    411555      ******
** DBA.dupowned2.ASIQ_IDX_T1708_C1_FP    411555      ******

Partition Summary
   Database Objects Checked                          2
   Blockmap Identity Count                           2
   Bitmap Count                                      2
======================== ==============================
. . .
```

This section of the report names all objects involved in conflicts in which the same physical block on disk is allocated to more than one database object. The report provides the physical block numbers involved in these conflicts.

This section of the report can help you recover from corruption caused by block ownership conflicts. In the event of a block ownership conflict, contact SAP Support for advice on how to resolve the reported conflicts.

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 IQ demonstration database `iqdemo`.

See *sp_iqcheckdb Procedure* and *DBCC_LOG_PROGRESS Option* in *SAP IQ SQL Reference* for syntax and usage information.

6.3.3 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>` is greater than 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 percent of cache. See *SAP IQ SQL Reference*.

6.4 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.

In this section:

Analysis of Index Errors [page 82]
Use `sp_iqcheckdb` to analyze index inconsistencies.

Index Error Repair [page 85]
Use the `sp_iqrebuildindex` procedure to repair an index, then run `sp_iqcheckdb` in verify mode to check for index inconsistencies.

Analysis of Allocation Problems [page 85]
Use `sp_iqcheckdb` to analyze allocation problems.

Repairing Allocation Problems using DBCC [page 87]
Use `sp_iqcheckdb dropleaks` to repair database allocation problems.

6.4.1 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.

DBCC Message	Description
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.
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.

6.4.2 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.

> **i** Note
>
> The `sp_iqrebuildindex` procedure cannot repair FP indexes. SAP IQ has no functionality to repair FP indexes.

6.4.3 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 9: 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.

6.4.4 Repairing Allocation Problems using DBCC

Use `sp_iqcheckdb dropleaks` to repair database allocation problems.

Context

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

Procedure

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
   ```

   > **i** 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.

Results

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

Example

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
```

6.5 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.

In this section:

Before Forced Recovery [page 89]
Restricting database access provides greater control over inadvertent database opens during forced recovery.

Starting a Server in Forced Recovery Mode [page 90]
Forced recovery allows the server to start if the allocation map is inconsistent.

Recovering Leaked Space [page 92]
Use the `sp_iqcheckdb` stored procedure in `dropleaks` mode to recover leaked storage space within the specified database.

6.5.1 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 the following, just after you start the connection where you are performing forced recovery:

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

Then perform the following on the same connection after recovery:

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

The disadvantage is that this method precludes emergency access from another DBA connection.

6.5.2 Starting a Server in Forced Recovery Mode

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

Forced Recovery on Multiplex Servers

Context

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 SAP SQL Anywhere errors, such as SA transaction log replay failure.

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

Procedure

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).

Results

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

Example

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.

Caution

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.

Forced Recovery on Simplex Servers

Context

If you cannot start a simplex server or database, 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 SAP SQL Anywhere errors, such as SA transaction log replay failure.

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

Procedure

1. Start the server with the `-iqfrec` flag:

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

2. Connect to the server and run:

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

3. Correct errors and rerun `sp_iqcheckdb`. Repeat until no errors result.

4. Shut down and restart the server normally (without the flags in Step 2).

Results

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

Example

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.

> ⚠ Caution
>
> 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.

6.5.3 Recovering Leaked Space

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

Context

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.

Procedure

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.

6.6 Problems Reported by DBCC

Messages are reported for problems that DBCC cannot repair.

Table 10: Messages for Problems DBCC Cannot Repair

DBCC Message	Description
FP Lookup Table Inconsistencies	An unrepairable error, where the 1-byte or 2-byte FP is internally inconsistent.

DBCC Message	Description
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.

In this section:

Index Problems that DBCC Cannot Repair [page 94]
Use these suggestions to repair inconsistent indexes.

Dropping Inconsistent Indexes, Tables, or Columns [page 95]
Use these suggestions to resolve issues with unrepairable indexes, columns, or tables.

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

6.6.2 Dropping Inconsistent Indexes, Tables, or Columns

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

Context

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.

> **i Note**
>
> You should not attempt to force drop objects unless SAP 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.

> **i 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.

Procedure

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 *SAP IQ Installation and Update 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.

6.7 DBCC Error Messages

These are the most important messages in the DBCC output.

Table 11: DBCC Error Messages

DBCC Message	Description
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 rerun 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
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 Chunk-size 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 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.

DBCC Message	Description
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	An error in the hash or hash-range load or insertion assigned row IDs incorrectly from different hash partitions.
Hash Partition corruption, RID range mismatch	Unload and reload the table.

7 Backup Reference

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

In this section:

BACKUP DATABASE Statement [page 99]
Backs up an SAP IQ database on one or more archive devices.

RESTORE DATABASE Statement [page 105]
Restores an SAP IQ database backup from one or more archive devices.

sp_iqcheckdb Procedure [page 116]
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.

7.1 BACKUP DATABASE Statement

Backs up an SAP IQ database on one or more archive devices.
Quick Links:

Go to Parameters

Go to Examples

Go to Usage

Go to Standards

Go to Permissions

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>' }
   POINT IN TIME RECOVERY LOGS ONLY
```

```
   WITH COMMENT <comment>
dbspace-or-file - (back to backup-option)
   { DBSPACES identifier-list | FILES <identifier-list> | <archive-root> }
identifier-list - (back to dbspace-or-file)
   <identifier> [, … ]
archive-option - (back to Syntax)
   SIZE <integer> STACKER <integer>
```

Parameters

(back to top)

- **TO** specifies 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 IQ attempts with regard to output devices.
- **WITH COMMENT** specifies an optional comment recorded in the archive file and in the backup history file. Maximum length is 32 KB. 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 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 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** specifies the number of blocks to write at one time. The value must be greater than 0, or SAP 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 – specifies a full backup; all blocks in use in the database are saved to the archive devices. This is the default action.
  - INCREMENTAL – specifies 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 – specifies an incremental backup; all blocks changed since the last full backup are saved to the archive devices.

- **VIRTUAL DECOUPLED** specifies a decoupled virtual backup. For the backup to be complete, you must copy the SAP IQ dbspaces after the decoupled virtual backup finishes, and then perform a nonvirtual incremental backup.
- **VIRTUAL ENCAPSULATED** specifies 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.
- **POINT IN TIME RECOVERY LOGS ONLY** backs up the point-in-time (PITR) logs to the dbspace set with the ALTER DBSPACE IQ_SYSTEM_LOG RENAME statement:

  ```
  BACKUP DATABASE
  POINT IN TIME RECOVERY LOGS ONLY TO ' PITR-archive-directory '
  ```

  The PITR archive directory is set with the ALTER DBSPACE IQ_SYSTEM_LOG RENAME statement.

  POINT IN TIME RECOVERY LOGS ONLY supports only one TO clause, which must point to the PITR archive directory. No other options are allowed.
- **SIZE** specifies 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 1 GB is inappropriate. For example, for a 3.5 GB 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.

  If a size less than 1 GB is specified, a SIZE warning message appears. The backup proceeds but uses the minimum default file size instead of the specified value. For example, if you specify a file size of 1000000 KB, a default file size of 2 GB (UNIX) or 1.5 GB (Windows) is used instead.

Platform	Default SIZE for Tape	Default SIZE for Disk
UNIX	none	2 GB (2097152 KB)
Windows	1.5 GB SIZE must be a multiple of 64. Other values are rounded down to a multiple of 64.	1.5 GB (1572864 KB)

  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** specifies 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 IQ waits 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)

Example 1

(UNIX) this example backs up the `iqdemo` database onto tape devices `/dev/rmt/0` and `/dev/rmt/2` on an Oracle 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
```

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

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)

The SAP 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 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 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 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 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 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 IQ system tables.

`BACKUP DATABASE` and `RESTORE DATABASE` write your SAP 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 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.

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

For the SAP 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 IQ assumes that each created disk file is no larger than 2 GB on UNIX, or 1.5 GB on Windows.

An example of an archive device for the SAP 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. Although Windows supports tape partitioning, SAP 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.

  > **⚠ Caution**
  >
  > For backup (and for most other situations) SAP 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 IQ interprets this name as a disk file name.
- SAP 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 IQ does rewind tapes after using them on rewinding devices.
- During backup and restore operations, if SAP 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)

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- SAP Database Products—Not supported by SAP ASE.

Permissions

(back to top)

Requires one of:

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

7.2 RESTORE DATABASE Statement

Restores an SAP IQ database backup from one or more archive devices.
Quick Links:

Go to Parameters

Go to Examples

Go to Usage

Go to Standards

Go to Permissions

Syntax

Syntax 1

```
RESTORE DATABASE '<db_file>'
    [MULTIPLEX]'<archive_device>' [ FROM '<archive_device>' ]…
   … [ CATALOG ONLY ]
   … [ KEY <key_spec> ]
   … [ [ RENAME <logical-dbfile-name> TO '<new-dbspace-path>']...
       | relative or absolute path of the database to be restored. Can be the
                          original location, or a new location for the catalog
store file.VERIFY [ COMPATIBLE ] ]
```

Syntax 2

```
RESTORE DATABASE '<database-name>'
   [ restore-option ...]
```

```
   FROM '<archive_device>' <...>

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

Syntax 3

```
RESTORE DATABASE '<db_file>'
  FROM '<file-name>' <…>
  USING LOG PATH '<directory>', […]
  RECOVER UNTIL  [ <TIMESTAMP>  <timestamptz> | <OFFSET> <log-offset> ]
  ON TIMELINE '<GUID>'
 [ OVERWRITE EXISTING CLEAR LOG ]
KEY <key_spec>
[ VERIFY COMPATIBLE ]
```

Parameters

(back to top)

- **db_file**
- **MULTIPLEX** specifies that you are restoring a shared-nothing multiplex. All secondary nodes with primary or mirror DAS dbfiles must be started before you issue a separate RESTORE command with the MULTIPLEX keyword. All secondary nodes with primary DAS dbfiles search for `<archive_device>.<server-id>`.
- **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 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'
  ```

  For 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.

  > **i** Note
  >
  > Only certain third-party products are certified with SAP 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 IQ database, make sure it is certified. See the *Release Bulletin*, or see the SAP IQ Certification Reports for the SAP IQ product in Technical Documents.

For the SAP 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 IQ API DLL that specifies a nonrewinding tape device on a UNIX-like operating system is:

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

- **USING LOG PATH directory** instructs server to search along multiple paths for point-in-time recovery logs:

```
USING LOG PATH ' <directory 1>,  <directory 2>, ... '
```

Use a comma as a delimiter between directory names. The log name does not need to be specified. If any required files are missing, the server reports an error.

On multiplex servers use the current transaction log from the coordinator node. Do not include transaction logs from secondary nodes. Including transaction logs from secondary nodes causes point-in-time recovery to fail, and return a `Files are missing for Point-in-time-Recovery` error.

- **RECOVER UNTIL . . .** recovers data from the recovery logs up to the date and time specified by the timestamp, or transaction log-offset:

```
RECOVER UNTIL [ TIMESTAMP <timestamptz> | OFFSET <log-offset> ]
```

`<timestamptz>` is a TIMESTAMP WITH TIMEZONE data type. `<logoffset>` is an UNSIGNED BIGINT that represents a transaction log offset.

Always specify a timestamp that is greater than the backup time of the data backup specified in the FROM clause of the restore command. This ensures that the database includes all committed transactions in the recovery logs. If the specified point in time is earlier than the last checkpoint in the backup database, the server returns an error.

! Restriction

- A dbspace that you create after enabling point-in-time recovery may only be recorded in the point-in-time recovery log. If this is the case, you cannot rename the dbspace during a RESTORE DATABASE RECOVER UNTIL operation.
- If RESTORE DATABASE cannot locate the transaction log, RLV log, and point-in-time recovery logs during a point-in-time recovery, the recovery operation fails. This can happen in cases when the database is restored to a new location and there are no logs available in the new environment. In cases like this, use the CLEAR LOG clause to ignore the current log area of the database you want to restore and cancel the automatic log backups that normally occur during a recovery operation.

- **ON TIMELINE** restores the database to an alternate timeline:

```
ON TIMELINE '<GUID>'
```

The ON TIMELINE clause requires a timeline GUID that identifies the alternate timeline. Point-in-time recovery operations without this clause restore to the current timeline.

- **OVERWRITE EXISTING** overwrites existing dbfiles and transaction logs during a point-in-time recovery operation:

  ```
  OVERWRITE EXISTING
  ```

  Point-in-time recovery operations generally restore dbfiles to a different location than the current dbspace. Use the OVERWRITE EXISTING clause to restore a database to a location that already has a database, and overwrite any existing dbfiles with the same name.

  The OVERWRITE EXISTING clause can appear as a restore option at any position in a RESTORE DATABASE statement.
- **CLEAR LOG** ignores the current log area of the database during a restore: cancels automatic log backups during a point-in-time recovery operation:

  ```
  CLEAR LOG
  ```

  If RESTORE DATABASE cannot locate the transaction log, RLV log, and point-in-time recovery logs during a point-in-time recovery, the recovery operation fails. This can happen in cases when the database is restored to a new location and there are no logs available in the new environment. In cases like this, use the CLEAR LOG clause to ignore the current log area of the database you want to restore and cancel the automatic log backups that normally occur during a recovery operation.

  During point-in-time recovery, the restore automatically backs up the existing transaction log and RLV log. Every point-in-time restore looks for the existing transaction log, RLV log, and point in time logs in the existing database directories. The restore operation returns an error if it fails to locate these logs. If you want to restore to a new environment, however, and have no existing logs to back up, supply a RESTORE command with the CLEAR LOG clause to stop SAP IQ from seeking existing log files in the database directories.
- **CATALOG ONLY** restores only the backup header record from the archive media. Cannot be used with the MULTIPLEX keyword.
- **RENAME** restore one or more SAP 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.

  If the dbspace name contains a file extension such as `.iq` or `.iqtmp`, you must enclose the dbspace name in double quotation marks when specifying the name in a `RESTORE DATABASE` command `RENAME` clause. For example:

  ```
  RENAME temp1 TO '/work/temp1_res.iqtmp.iqtmp'
  ```

```
DBSPACENAME "temp1_res.iqtmp"
```

Or:

```
RENAME "test_prod2.iq" TO '/test/test_prod2.iq'
```

- **VERIFY [COMPATIBLE]** directs the server to validate the specified SAP IQ database backup archives for a full, incremental, incremental since full, or virtual backup. The backup must be SAP 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.

  > **i** 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)

Example 1

this UNIX example restores the `iqdemo` database from tape devices `/dev/rmt/0` and `/dev/rmt/2`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` commands for the type of backup, that, use the naming convention appropriate for your UNIX platform. (Windows does not support this feature.) 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

```
                database is ready to be used. The database is left in the
state that existed at theRESTORE 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
```

Example 5

Point-in-time recovery using point-in-time recovery logs only:

```
// enable Point in time recovery
SET OPTION PUBLIC.IQ_POINT_IN_TIME_RECOVERY_LOGGING = 'ON'
ALTER DBSPACE IQ_SYSTEM_LOG RENAME '/demo/pitrLog/PITRLOG'
BACKUP DATABASE FULL to '/demo/dataBackup/FULL1'
// perform some DDL/DML operations
CREATE TABLE T1 ( ID INT )
INSERT INTO T1 VALUES (1)
// record timestamp TS1 / OFFSET OFS1
TS1: select getdate()
OFS1: select db_property('LastCommitRedoPos')
// perform DDL/ DML operations
CREATE TABLE T2 ( ID INT )
INSERT INTO T2 VALUES (1)
// record timestamp TS2 / OFFSET OFS2
TS2: select getdate()
OFS2: select db_property('LastCommitRedoPos')
// restore the database to point in time TS1 / OFS1
// shut down the database
// start a utility server
RESTORE DATABASE 'iqdemo' FROM '/demo/dataBackup/FULL1'
    RECOVER UNTIL TIMESTAMP  '<TS1>'
    USING LOG PATH '/demo/pitrLog/' ,
OVERWRITE EXISTING
// or
RESTORE DATABASE 'iqdemo' FROM '/demo/dataBackup/FULL1'
    RECOVER UNTIL OFFSET <OFS1>
    USING LOG PATH '/demo/pitrLog/' OVERWRITE EXISTING
```

Example 6

Point-in-time recovery using point-in-time recovery logs and point-in-time recovery log backup archives:

```
// enable point-in-time recovery
SET OPTION PUBLIC.IQ_POINT_IN_TIME_RECOVERY_LOGGING = 'ON
ALTER DBSPACE IQ_SYSTEM_LOG RENAME '/demo/pitrLog/PITRLOG'
BACKUP DATABASE FULL to '/demo/dataBackup/FULL1'
// perform DDL / DML operations
CREATE TABLE T1 ( ID INT )
INSERT INTO T1 VALUES (1)
// record timestamp TS1 / OFFSET OFS1
TS1: select getdate()
OFS1: select db_property('LastCommitRedoPos')
// perform DDL / DML operations
CREATE TABLE T2 ( ID INT )
INSERT INTO T2 VALUES (1)
// record timestamp TS2 / OFFSET OFS2
TS2: select getdate()
OFS2: select db_property('LastCommitRedoPos')
// perform a PITR log backup
BACKUP DATABASE
    POINT IN TIME RECOVERY LOGS ONLY TO '/demo/pitrLogBackup/PITR1'
// perform DDL/ DML operations
CREATE TABLE T3 ( ID INT )
INSERT INTO T3 VALUES (1)
```

```
// restore the database to point-in-time TS2 OR OFS2
// shutdown the database
// start a utility server
RESTORE DATABASE 'iqdemo'
    FROM '/demo/dataBackup/FULL1'
    RECOVER UNTIL TIMESTAMP  '<TS2>'
    USING LOG PATH '/demo/pitrLogBackup/', '/demo/pitrLog/'
    OVERWRITE EXISTING
// or
RESTORE DATABASE 'iqdemo'
    FROM '/demo/dataBackup/FULL1'
    RECOVER UNTIL OFFSET <OFS2>
    USING LOG PATH '/demo/pitrLogBackup/', '/demo/pitrLog/'
    OVERWRITE EXISTING
```

Example 7

Point-in-time recovery on an RLV-enabled database with an incremental backup.

```
// perform a full database backup
BACKUP DATABASE FULL TO '/demo/dataBackup/FULL1'
// perform DDL / DML operations
CREATE TABLE T0 ( ID INT )
INSERT INTO T0 VALUES (1)
RECORD TIME STAMP TS0 / OFFSET OFS0
// enable point-in-time recovery using an incremental backup
SET OPTION PUBLIC.IQ_POINT_IN_TIME_RECOVERY_LOGGING = 'ON
ALTER DBSPACE IQ_SYSTEM_LOG RENAME '/demo/pitrLog/PITRLOG'
BACKUP DATABASE INCREMENTAL to '/demo/dataBackup/INCR1'
// perform DDL / DML operations
CREATE TABLE T1 ( ID INT )
INSERT INTO T1 VALUES (1)
// record timestamp TS1 / OFFSET OFS1
TS1: select getdate()
OFS1: select db_property('LastCommitRedoPos')
// perform DDL / DML operations
CREATE TABLE T2 ( ID INT )
INSERT INTO T2 VALUES (1)
// record timestamp TS2 / OFFSET OFS2
//perform a PITR log backup
BACKUP DATABASE
    POINT IN TIME RECOVERY LOGS ONLY TO '/demo/pitrLogBackup/PITR1'
// note the RLV DBspace files and their location
select * from sp_iqfile() where segmentType = 'RLV'"
// restore the database to point-in-time TS1 OR OFS1
// shutdown the database
// Do a physical file backup of all files in the RLV Dbspace.
// start a utility server
// Perform a restore of database from backup FULL1
RESTORE DATABASE 'iqdemo'
    FROM '/demo/dataBackup/FULL1'
    OVERWRITE EXISTING
// restore the RLV Dbspace files from the backup taken above to its original
location
// perform a point-in-time recovery to time TS1 using incremental backup INCR1
RESTORE DATABASE 'iqdemo'
    FROM '/demo/dataBackup/INCR1'
    RECOVER UNTIL TIMESTAMP  '<TS1>'
    USING LOG PATH '/demo/pitrLogBackup/PITR1/'
    OVERWRITE EXISTING CLEAR LOG
```

Example 8

Point-in-time recovery on multiplex:

```
// enable point-in-time recovery
SET OPTION PUBLIC.IQ_POINT_IN_TIME_RECOVERY_LOGGING = 'ON
ALTER DBSPACE IQ_SYSTEM_LOG RENAME '/demo/pitrLog/PITRLOG'
// shut down all servers
// restart coordinator
// connect to coordinator and take a full backup
BACKUP DATABASE FULL to '/demo/dataBackup/FULL1'
// synchronize all secondary nodes & then start them
// perform DDL / DML operations in coordinator
// connect to coordinator
CREATE TABLE T1 ( ID INT );
INSERT INTO T1 VALUES (1);
COMMIT;
// record timestamp TS1 / OFFSET OFS1;
TS1: select getdate()
OFS1: select db_property('LastCommitRedoPos')
// perform DDL / DML operations on writers
// connect to Writer 2
CREATE TABLE T2 ( ID INT );
INSERT INTO T2 VALUES (1);
COMMIT;
// record timestamp TS2 / OFFSET OFS2;
TS2: select getdate()
OFS2: select db_property('LastCommitRedoPos')
// connect to writer3
INSERT INTO T1 VALUES(2);
INSERT INTO T2 VALUES(2);
COMMIT;
// record timestamp TS3 / OFFSET OFS3;
TS3: select getdate()
OFS3: select db_property('LastCommitRedoPos')
// perform a PITR log backup
BACKUP DATABASE
POINT IN TIME RECOVERY LOGS ONLY TO '/demo/pitrLogBackup/PITR1'
// perform DDL/ DML operations
// Connect to Writer3
CREATE TABLE T3 ( ID INT );
INSERT INTO T3 VALUES (1);
COMMIT;
// record timestamp ts4 / offset ofs4;
TS4: select getdate()
OFS4: select db_property('LastCommitRedoPos')
// restore the database to point-in-time TS3 OR OFS3
// shut down the database
// start a utility server and perform point-in-time
// recovery for the coordinator node
RESTORE DATABASE 'iqdemo'
FROM '/demo/dataBackup/FULL1'
RECOVER UNTIL TIMESTAMP  '<TS3>'
USING LOG PATH '/demo/pitrLogBackup/', '/demo/pitrLog/'
OVERWRITE EXISTING
// or
RESTORE DATABASE 'iqdemo'
FROM '/demo/dataBackup/FULL1'
RECOVER UNTIL OFFSET <OFS3>
USING LOG PATH '/demo/pitrLogBackup/', '/demo/pitrLog/'
OVERWRITE EXISTING
```

Example 9

Re-enabling point-in-time recovery logging after a multiplex failover. In this scenario, writer 1 becomes the coordinator after the failover:

```
// enable point-in-time recovery
SET OPTION PUBLIC.IQ_POINT_IN_TIME_RECOVERY_LOGGING = 'ON
ALTER DBSPACE IQ_SYSTEM_LOG RENAME '/demo/pitrLog/PITRLOG2'
// Shut down all servers
```

```
// Restart coordinator
BACKUP DATABASE FULL to '/demo/dataBackup/FULL1'
// Synchronize then start all secondary nodes
```

Usage

(back to top)

The RESTORE DATABASE command requires exclusive access on an Oracle Solaris platform. On Solaris, 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 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 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. Avoid starting the database until after the entire sequence of incremental backups has completed.

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).

> **i Note**
>
> If a transaction log with the same name already exists in the target location, RESTORE DATABASE generates an exception and does not restore your files. Fix the problem by using one of these two workarounds:
>
> - Rename the transaction log file of the source database using the dblog database administration utility.
> - Rename the transaction log file in the new location using the RENAME clause of the RESTORE DATABASE statement.

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 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 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 IQ data before starting the restore.
- During backup and restore operations, if SAP 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)

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- SAP Database Products—Not supported by SAP ASE.

Permissions

(back to top)

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.

7.3 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.

> **i** Note
>
> `sp_iqcheckdb` is the user interface to the SAP 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 [nomirrorcheck]
   | 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 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.

nomirrorcheck

Applies to shared-nothing multiplex. Excludes mirror DAS dbfiles from database consistency checking.

index-type

One of the following index types: `FP`, `CMP`, `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.

> **i 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.

> **i** 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. You must also have 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.

> **i** Note
>
> See Database Verification [page 74] for detailed information on checking database consistency with `sp_iqcheckdb`.

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).
	
	i Note
	
	If `sp_iqcheckdb` detects a block ownership conflict, it adds a `**Blocks with Multiple Owners***` section to the report, listing the implicated object names, and physical block numbers. Block ownership conflicts are only analyzed if the target of `sp_iqcheckdb` is either a database or a dbspace. Example of a block ownership conflict:
	
	`** Blocks with Multiple Owners ******`
	`** DBA.dupowned1.ASIQ_IDX_T1707_C1_FP 411552 ******`
	`** DBA.dupowned2.ASIQ_IDX_T1708_C1_FP 411552 ******`
	`** DBA.dupowned1.ASIQ_IDX_T1707_C1_FP 411553 ******`
	`...`
	
	This section of the report can help you recover from corruption caused by block ownership conflicts. In the event of a block ownership conflict, contact SAP Support for advice on how to resolve the reported conflicts.
	
	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.
	
	i 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 *SAP IQ Administration: Multiplex*. `resetclocks` corrects the values of internal data-

Mode	Description
	base versioning clocks, in the event that these clocks are behind. Do not use the `resetclocks` option for any other purpose, unless you contact SAP 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'`
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. **i Note** `sp_iqcheckdb` does not check referential integrity or repair referential integrity violations.
Dropleaks	When the SAP 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.

> **i Note**
>
> If `sp_iqcheckdb` cannot access DAS dbfiles in your DAS dbspace, DBCC will issue error messages immediately and will not retry the DAS dbspace check. Make sure all DAS dbfile nodes and mirror DAS dbfile nodes are running before executing `sp_iqcheckdb`.

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.

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

Checks the allocation for the entire database:

```
sp_iqcheckdb 'allocation database'
```

Executes dropleaks mode on dbspace IQ_SYSTEM_MAIN:

```
sp_iqcheckdb 'dropleaks dbspace IQ_SYSTEM_MAIN'
```

Performs 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'
```

Checks all indexes of the type `FP` in the database:

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

Verifies the `FP` and `HG` indexes in the table `t1` and the `HNG`indexes in the table `t2`:

```
sp_iqcheckdb 'verify indextype FP indextype HG table t1 indextype HNG 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.

> **i 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
```

> **i** 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.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

www.sap.com/contactsap

© 2019 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see https://www.sap.com/about/legal/trademark.html for additional trademark information and notices.

THE BEST RUN SAP