IBM solidDB
IBM solidDB Universal Cache
Version 6.3

Administrator Guide

SC23-9824-03

Note
Before using this information and the product it supports, read the information in "Notices" on page 277.

First edition, third revision

This edition applies to version 6, release 3 of IBM solidDB (product number 5724-V17) and IBM solidDB Universal Cache (product number 5724-W91) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

Summary of changes

Changes for revision 03

- Section ADMIN COMMAND updated with the following changes:

  The following undocumented ADMIN COMMAND 'trace' options have been added:

  – est - SQL estimator information
  – estplans - SQL execution plan
  – flow - advanced replication statements
  – rexec - remote procedure call information
  – batch - background job and deferred procedure call information

  The following undocumented ADMIN COMMANDs have been added:

  – 'errormessage <string>' – Outputs the user-defined `<string>` to the error message log (`solerror.out`).
  – 'logmessage <string>' – Outputs the user-defined `<string>` to the message log (`solmsg.out`).
  – 'tracemessage <string>' – Outputs the user-defined `<string>` to the trace message log (`soltrace.out`).
- Information on the support of **IndexFile.DirectIO** and **Logging.DirectIO** parameters added in section Server-side configuration parameters: these parameters are not effective in Windows® environments; in Windows environments, database files always use Direct I/O.
- Section Appendix D, “Error codes,” on page 179 updated to correspond to `ADMIN COMMAND 'errorcode all'` output. Previously undocumented messages added in the following sections:
  – “solidDB API errors” on page 219
  – “solidDB AT (timed commands) messages” on page 241
  – “solidDB BCKP (backup) messages” on page 241
  – “solidDB COM (communication) messages” on page 237
  – “solidDB CP (checkpoint) messages” on page 241
  – “solidDB DBE (database engine) errors and messages” on page 239
  – “solidDB FIL (file system) messages” on page 246
  – “solidDB HSB (HotStandby) errors and messages” on page 243
  – “solidDB INI (configuration file) messages” on page 242
  – “solidDB LOG (logging) messages” on page 242
  – “solidDB RPC errors and messages” on page 219
  – “solidDB SNC (synchronization) messages” on page 245
  – “solidDB SRV (server) errors” on page 238
  – “solidDB TAB (table) messages” on page 247
  – “solidDB XS (external sorter) errors and messages” on page 246
- New section added: “Viewing error message descriptions with `ADMIN COMMAND 'errorcode'`” on page 16

Changes for revision 02

- New section added: “Setting up database environment” on page 12

- Section “Encrypting a database” on page 43 updated.
- The following new parameters have been added in “Srv section” on page 156:
  - **Srv.HealthCheckEnabled**
  - **Srv.HealthCheckInterval**
  - **Srv.HealthCheckTimeout**
- New parameter **HotStandby.TCConnect** added in “HotStandby section” on page 136.
- New parameter **LogReader.Silent** added in “LogReader section” on page 146.
- New parameter **SQL.DecFloatPrecision16** added in “SQL section” on page 152.
- New `ADMIN COMMAND 'indexusage'` added in section “ADMIN COMMAND” on page 257.

Changes for revision 01

- New section added: “Running solidDB as a Windows service” on page 38.
- Factory value for parameter **Srv.MemoryReportLimit** changed from 100 MB to 0 (no reporting) in section “Srv section” on page 156.
- New server error, 30152 (Memory allocation size has exceeded a given value), added in section “solidDB server errors” on page 210.
- New option, STOPPING, added for `ADMIN COMMAND 'status backup | netbackup'` in *Appendix solidDB® ADMIN COMMAND Syntax*, section “ADMIN COMMAND” on page 257.
- The syntax for the start command for solidDB Export (solexp) clarified in section “Starting solidDB Export” on page 77.
- Communication protocol Shared Memory (shmem) is deprecated as of release 6.3 Fix Pack 1. Examples using *shmem* updated throughout the manual: instead of *shmem <servername>, tcpip 1964* is used.
- Default database block size corrected to 16 KB in section “Setting database block size (BlockSize) and location (FileSpec)” on page 14.
- Command for stopping monitoring (`ADMIN COMMAND 'pmon diff stop'`) corrected in section “Producing a continuous performance monitoring report” on page 20.
- Factory value for parameter **MME.LockEscalationEnabled** corrected in section “MME section” on page 148: default is '**no**'.

About this manual

IBM® solidDB® is a versatile database management system that can be used in systems starting from small embedded systems to large-scale systems. Various functional IBM solidDB components may be enacted to serve special needs. Such components are:

- an in-memory database
- a highly available hot standby configuration
- advanced asynchronous replication
- a library for directly linking applications with the server code.

All of the above mentioned components are orthogonal, that is they can be used in the presence of other components. An administrator of solidDB can use a wide range of configuration options and tools to set up the product in the most appropriate way.

This guide describes how to set up, monitor, manage, and optimize the basic database server function of the product. More detailed information about configuring specific solidDB components are included in the related manuals.

This guide assumes the reader has general DBMS knowledge and a familiarity with SQL.

Typographic conventions

solidDB documentation uses the following typographic conventions:

Table 1. Typographic conventions

Format	Used for
Database table	This font is used for all ordinary text.
NOT NULL	Uppercase letters on this font indicate SQL keywords and macro names.
`solid.ini`	These fonts indicate file names and path expressions.
`SET SYNC MASTER YES;` `COMMIT WORK;`	This font is used for program code and program output. Example SQL statements also use this font.
run.sh	This font is used for sample command lines.
TRIG_COUNT()	This font is used for function names.
java.sql.Connection	This font is used for interface names.
LockHashSize	This font is used for parameter names, function arguments, and Windows registry entries.
argument	Words emphasized like this indicate information that the user or the application must provide.

Table 1. Typographic conventions (continued)

Format	Used for
Administrator Guide	This style is used for references to other documents, or chapters in the same document. New terms and emphasized issues are also written like this.
File path presentation	Unless otherwise indicated, file paths are presented in the UNIX® format. The slash (/) character represents the installation root directory.
Operating systems	If documentation contains differences between operating systems, the UNIX format is mentioned first. The Microsoft® Windows format is mentioned in parentheses after the UNIX format. Other operating systems are separately mentioned. There may also be different chapters for different operating systems.

Syntax notation conventions

solidDB documentation uses the following syntax notation conventions:

Table 2. Syntax notation conventions

Format	Used for
`INSERT INTO table_name`	Syntax descriptions are on this font. Replaceable sections are on *this* font.
`solid.ini`	This font indicates file names and path expressions.
[ ]	Square brackets indicate optional items; if in bold text, brackets must be included in the syntax.
|	A vertical bar separates two mutually exclusive choices in a syntax line.
{ }	Curly brackets delimit a set of mutually exclusive choices in a syntax line; if in bold text, braces must be included in the syntax.
...	An ellipsis indicates that arguments can be repeated several times.
. . .	A column of three dots indicates continuation of previous lines of code.

1 Managing data with solidDB

The core of solidDB is a relational database server. This database server accepts queries in the SQL language. Usually, these SQL queries are submitted by a "client" application that sends SQL statements to the server and receives result sets back from the server.

In addition, solidDB has synchronization features that allow updated data in one solidDB to be sent to one or more other solidDBs. solidDB also allows you to run a pair of solidDBs in a hot standby configuration, and link your client application directly to the database server routines for higher performance and tighter control over the server. These features, called HotStandby and linked library access, are described later in this chapter.

This chapter describes the underlying components and processes that make solidDB the solution for managing distributed data in today's complex distributed system environments. It provides you with the background necessary to administer and maintain solidDB in your network environment.

solidDB data management components

solidDB includes the components described in the following sections.

Programming interfaces (ODBC and JDBC)

To submit a query (an SQL statement) to a database server, a client must be able to communicate with that database server. solidDB, like many other database servers, uses "drivers" to enable this communication. Client applications call functions in the driver, and the driver then handles the communications and other details with the server. For example, you might write a C program that calls functions in the ODBC driver, or you might write a Java™ program that calls functions in the JDBC driver.

ODBC

solidDB provides ODBC and JDBC drivers that communicate with solidDB. The solidDB ODBC Driver conforms to the Microsoft ODBC 3.51 API standard. solidDB ODBC Driver supported functions are accessed with solidDB ODBC API, a Call Level Interface (CLI) for solidDB databases, which is compliant with ANSI X3H2 SQL CLI.

JDBC

The solidDB JDBC Driver allows Java applications to access the database by using JDBC. The solidDB JDBC Driver implements most of the JDBC 2.0 specification.

Proprietary interfaces

solidDB also provides proprietary interfaces. These allow, for example, C programs to directly call functions inside the database server. These proprietary interfaces are provided with the solidDB linked library access (described later).

For more details on ODBC, JDBC, and solidDB's propriety programming interfaces, see the *IBM solidDB Programmer Guide*.

Network communications layer

solidDB runs on all major network types and supports all of the main communication protocols (such as TCP/IP). Developers can create distributed applications for use in heterogeneous computing environments. Read 6, "Managing network connections," on page 99, for more details on network communication.

SQL parser and optimizer

solidDB uses SQL syntax based on the ANSI X3H2 and IEC/ISO 9075 SQL standards. SQL-89 Level 2 standard is fully supported as well as SQL-92 Entry Level. Many features of full SQL-92 and SQL-99 standards are also supported. solidDB contains an advanced cost-based optimizer, which ensures that even complex queries can be run efficiently. The optimizer automatically maintains information about table sizes, the number of rows in tables, the available indices, and the statistical distribution of the index values.

See the section "solidDB SQL Optimizer" on page 5 for more details on the solidDB SQL Optimizer.

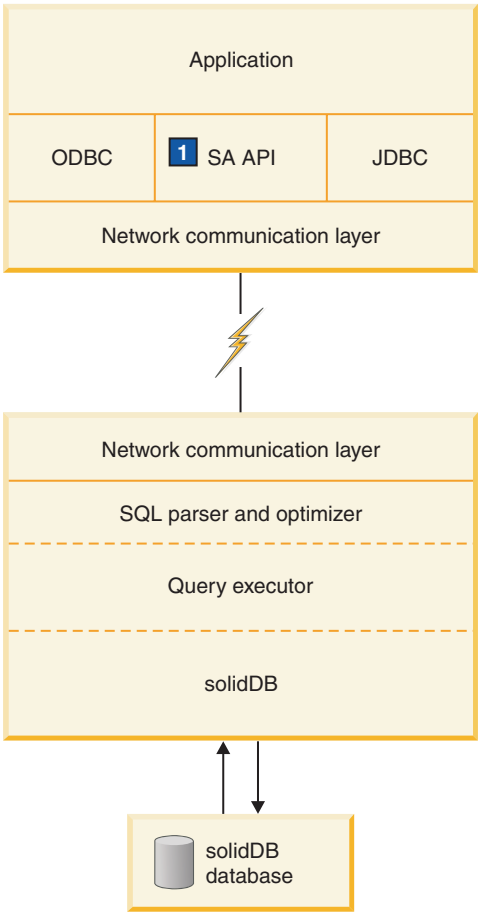
Optimizer hints

Optimizer hints (which are an extension of SQL) are directives specified through embedded pseudo comments within query statements. The optimizer detects these directives or hints and bases its query execution plan accordingly. Optimizer hints allow applications to be optimized under various conditions to the data, query type, and the database. They not only provide solutions to performance problems occasionally encountered with queries, but shift control of response times from the system to the user.

For more details on optimizer hints, read *IBM solidDB SQL Guide*.

solidDB

The solidDB processes the data requests submitted via solidDB SQL. The solidDB server shown in Figure 1 on page 3 stores data and retrieves it from the database.



1. SA API is solidDB's own API for use with the accelerator library. For details, see *IBM solidDB Linked Library Access User Guide*.

Figure 1. solidDB components

System tools and utilities

solidDB also includes the following tools for data management and administration:

Console tools

solidDB provides two ASCII-based console tools, solidDB Remote Control (solcon) and solidDB SQL Editor (solsql), to manage databases. These tools use a command line interface. Read 4, “Using solidDB data management tools,” on page 59 for details.

Tools for handling external ASCII data

solidDB provides the following tools for handling ASCII data:

- solidDB Speed Loader (solload) loads data from external ASCII files into a solidDB database. It is capable of inserting character data from character format. solidDB Speed Loader bypasses the SQL parser and uses direct writes to the database file with loading, which allows for fast loading speed.
- solidDB Export (solexp) writes from a solidDB database to character format files. It is capable of writing control files used by solidDB Speed Loader to perform data load operations.

- solidDB Data Dictionary (soldd) writes the data dictionary of a database. This tool produces a SQL script that contains data definition statements describing the structure of the database.

Read4, “Using solidDB data management tools,” on page 59, for details.

solidDB architecture

This section provides conceptual information that can give you an understanding in configuring solidDB to meet the needs of your own applications and platforms. It looks at the following:

- Data Storage
  - Main Storage Tree
  - Bonsai Tree Multiversioning and Concurrency Control
- Dynamic SQL Optimization
- Network Services
- Multithread processing

Data storage for disk-based tables

The main data structure used to store disk-based tables (D-tables) is a B-tree variation. The server uses two of these structures; the "main storage tree" holds permanent data, and a differential index tree called "Bonsai Tree" stores "new" data temporarily until it is ready to be moved to the main storage tree.

The internal part of the server taking care of storing D-tables is called the Disk-Based Engine (DBE).

Main storage tree

The main storage tree contains all the data in the server, including tables and indexes. Internally, the server stores ALL data in "indexes" — there are no separate tables. Each index contains either complete primary keys (i.e. all the data in a row) or secondary keys (what SQL refers to as "indexes" — just the column values that are part of the SQL index). There is no separate storage method for data rows, except for Binary Large Objects (BLOB) and other long column values.

All the indexes are stored in a single tree, which is the main storage tree. Within that tree, indexes are separated from each other by a system-defined index identification inserted in front of every key value. This mechanism divides the index tree into several logical index subtrees, where the key values of one index are clustered close to each other. For details on data clustering and primary key indexes, read the discussion of Primary Key Indexes in *IBM solidDB SQL Guide*.

solidDB Bonsai Tree multiversioning and concurrency control

The Bonsai Tree is a small active "index" (data storage tree) that efficiently stores new data (deletes, inserts, updates) in central memory, while maintaining multiversion information. Multiple versions of a row (old and new) can co-exist in the Bonsai Tree. Both the old and new data are used for concurrency control and for ensuring consistent read levels for all transactions without any locking overhead. With the Bonsai Tree, the effort needed for concurrency control is significantly reduced.

When a transaction is started, it is given a sequential Transaction Start Number (TSN). The TSN is used as the "read level" of the transaction; all key values inserted later into the database from other connections are not visible to searches

within the current transaction. This offers consistent index read levels that appear as if the read operation was performed atomically at the time the transaction was started. This guarantees read operations are presented with a consistent view of the data without the need for locks, which have higher overhead.

Old versions of rows (and the newer version(s) of those same rows) are kept in the Bonsai Tree for as long as there are transactions that need to see those old versions. After the completion of all transactions that reference the old versions, the "old" versions of the data are discarded from the Bonsai tree, and new committed data is moved from the Bonsai Tree to the main storage tree. The presorted key values are merged as a background operation concurrently with normal database operations. This offers significant I/O optimization and load balancing. During the merge, the deleted key values are physically removed.

Index compression

Two methods are used to store key values in the Bonsai Tree and the storage tree. First, only the information that differentiates the key value from the previous key value is saved. The key values are said to be prefix-compressed. Second, in the higher levels of the index tree, the key value borders are truncated from the end; that is, they are suffix-compressed.

Data storage for memory-based tables

solidDB allows for creation of memory-resident tables, so-called M-tables. The advantage of M-tables is their performance. M-tables have the same properties in terms of durability and recoverability as traditional disk-based tables (D-tables). The only difference is the location of the primary storage. M-tables are primarily stored in main memory, meaning that the bigger the in-memory database is, the more room it occupies in main memory. In addition to the actual data, the indexes for M-tables are built in main memory as well. solidDB uses a main-memory-optimized index technology called "tries" to implement the indexes. To evaluate the amount of memory needed to store the M-tables and their indexes, see the *IBM solidDB In-Memory Database User Guide*.

The internal part of the server taking care of storing M-tables is called Main-Memory Engine (MME)

solidDB SQL Optimizer

The solidDB SQL Optimizer, a cost-based optimizer, ensures that the execution of SQL statements is done efficiently. It uses the same techniques as a rules-based optimizer, relying on a preprogrammed set of rules to determine the shortest path to the results. For example, the SQL Optimizer considers whether or not an index exists, if it is unique, and if it is over single or composite table columns. However, unlike a rule-based optimizer, its cost-based feature can adapt to the actual contents of the database — for example, the number of rows and the value distribution of individual columns.

solidDB maintains the statistical information about the actual data automatically, ensuring optimal performance. Even when the amount and content of data changes, the optimizer can still determine the most effective route to the data.

Query processing

Query processing is performed in small steps to ensure that one time-consuming operation does not block another application's request. A query is processed in a sequence containing the following phases:

- Syntax analysis

- Creating the execution graph
- Processing the execution graph

Syntax analysis

An SQL query is analyzed and the server produces either a parse tree for the syntax or a syntax error. When a statement is parsed, the information necessary for its execution is loaded into the statement cache. A statement can be executed repeatedly without re-optimization, as long as its execution information remains in the statement cache.

Creating the execution graph

The execution graph, with the following features, is created from the query parse tree.

- Complex statements are written to a uniform and more simple form.
- If better performance will be realized, OR criteria are converted to UNION clauses. (For more details about OR vs. UNION, see the discussion of CONVERTORSTOUNIONS in the *IBM solidDB SQL Guide*.
- Intelligent join constraint transfer is performed to produce intermediate join results that reduce the join process execution time.

For details on each operation or unit in theexecution plan, read the discussion of the EXPLAIN PLAN FOR statement in the *IBM solidDB SQL Guide*.

Processing the execution graph

Processing of the execution graph is performed in three consecutive phases:

- Type-evaluation phase

  The column data types of the result set are derived from the underlying table and view definitions
- Estimate-evaluation phase

  The cost of retrieving first rows and also entire result sets is evaluated, and an appropriate search strategy is dynamically selected based on the parameter values that are bound to the statement.

  The SQL Optimizer bases cost estimates on automatically maintained information on key value distribution, table sizes, and other dynamic statistical data. No manual updates to the index histograms or any other estimation information is required.
- Row-retrieval phase

  The result rows of the query are retrieved and returned to the client application.

solidDB Network Services

solidDB Network Services are based on the remote procedure call (RPC) paradigm, which makes the communication interface simple to use. When a client sends a request to the server, it resembles calling a local function. The Network Services invisibly route the request and its parameters to the server, where the actual service function is called by the RPC Server. When the service function completes, the return parameters are sent back to the calling application.

In a distributed system, several applications may request a server to perform multiple operations concurrently. For maximum parallelism, solidDB Network Services use the operating system threads when available to offer a seamless multiuser support. On single-threaded operating systems, the Network Services extensively use asynchronous operations for the best possible performance.

Communication session layer

solidDB communication protocol DLLs (or static libraries) offer a standard internal interface to each protocol. The lowest part of the communication session layer works as a wrapper that takes care of choosing the correct protocol DLL or library that relates with the given address information. After this point, the actual protocol information of the session is hidden.

solidDB can listen to many protocols simultaneously.

Multithread processing

solidDB's multithread architecture provides an efficient way of sharing the processor within an application. A thread is a dispatchable piece of code that merely owns a stack, registers (while the thread is executing), and its priority. It shares everything else with all other active threads in a process. Creating a thread requires much less system overhead than creating a process, which consists of code, data, and other resources such as open files and open queues.

Threads are loaded into memory as part of the calling program; no disk access is therefore necessary when a thread is invoked. Threads can communicate using global variables, events, and semaphores.

If the operating system supports symmetric multi-threading between different processors, solidDB automatically takes advantage of the multiple processors.

Types of threads

The solidDB threading system consists of general purpose threads and dedicated threads.

General purpose threads

General purpose threads execute tasks from the server's tasking system. They execute such tasks as serving user requests, making backups, executing timed commands, merging indexes, and making checkpoints (storing consistent data to disk).

General purpose threads take a task from the tasking system, execute the task step to completion and switch to another task from the tasking system. The tasking system works in a round-robin fashion, distributing the client operations evenly between different threads.

The number of general purpose threads can be set in the `solid.ini` configuration file.

Dedicated threads

Dedicated threads are dedicated to a specific operation. The following dedicated threads may exist in the server:

- I/O manager thread

  This thread is used for intelligent disk I/O optimization and load balancing. All I/O requests go through the I/O manager, which determines whether to pass each I/O request to the cache or to schedule it among other I/O requests. I/O requests are ordered by their logical file address. The ordering optimizes the file I/O since the file addresses accessed on the disk are in close range, reducing the disk read head movement.
- Communication read threads

Applications always connect to a listener session that is running in the selector thread. After the connection is established, a dedicated read thread can be created for each client.

- One communication select thread per protocol (known as the selector thread)

  There is usually one communication selector thread per protocol. Each running selector thread writes incoming requests into a common message queue.
- Communication server thread (also known as the RPC server main thread)

  This thread reads requests from the common message queue and serves applications by calling the requested service functions.

2 Administering solidDB

This chapter describes how to maintain your solidDB installation. The administration tasks covered in this chapter are:

- Performing basic solidDB operations, such as starting and stopping the server
- Backing up the server
- Encrypting a database

Important: In the solidDB with linked library access, there are some differences in administration from standard solidDB. Wherever necessary, this chapter refers you to the *IBM solidDB Linked Library Access User Guide* for linked library access -specific information.

Background information

Using solidDB Embedded Engine databases 2.20 or prior

Beginning with solidDB Embedded Engine version 2.3 to the current version, the default collation sequence is set to the standard Latin-1. solidDB Embedded Engine databases that were created with version 2.20 or prior do not match the Latin-1 collation sequence. To convert the data to Latin 1 in a version 2.20 database, you must export the database from its tables, extract data definitions, and load the tables to the new database. For details, read “Tools sample: reloading a database” on page 80.

Special roles for database administration

solidDB has the following roles for administration and maintenance:

- SYS_ADMIN_ROLE

  This is the Database Administrator role and hasprivileges to all tables, indexes, and users, as well as theright to use solidDB Remote Control (solcon).This is also the role of the creator of the database.

- SYS_CONSOLE_ROLE

  This role has the right to use solidDB Remote Control,but has no other administration privileges.

- SYS_SYNC_ADMIN_ROLE

  This is an administration role for performingadministrative operations related to synchronization, such asdeleting messages. ("Messages" are used to pass informationback and forth between a master and its replicas. Forexample, to refresh the data that is in a master publication,the replica sends a REFRESH message, unless the synchronousrefresh mode is used.) Anyone with this accesshas all synchronization roles granted automatically. Thisrole automatically includes the SYS_SYNC_REGISTER_ROLE.

- SYS_SYNC_REGISTER_ROLE

  This is a role only for registering or unregistering areplica database to the master.

  You define these roles using the GRANT ROLE statement.For details, read "Managing User Privileges and Roles" in *IBM solidDB SQL Guide*.

Automated and manual administration

solidDB is designed for continuous, unattended operation and ease of deployment. It requires minimal maintenance. Administrative operations, including backups, can be performed programmatically using SQL extensions, which can run automatically or at an administrator's request.

Sometimes, however, it makes sense to administer systems manually. This chapter also refers you to the tools and methods available for performing manual administration. To perform administration tasks, you can issue solidDB SQL's own ADMIN COMMANDs in solidDB SQL Editor (solsql). For a comprehensive list of commands, refer to Appendix E, "solidDB ADMIN COMMAND syntax," on page 257.

If you are using solidDB with linked library access, the Control API gives a user application programmatic control over task execution. A Control API function is available for assigning priorities for such tasks as database backup, database checkpoint, and merge of the Bonsai Tree. The priority assignment determines in what order a task is run once it is executed. For details, read *IBM solidDB Linked Library Access User Guide*.

solidDB Remote Control (solcon) lets you enter administrative commands without using the ADMIN COMMAND syntax. See "solidDB Remote Control (solcon)" on page 59 for details.

Starting solidDB

Note:

This section applies to standard solidDB only. If you are using solidDB with linked library access, read the corresponding section in *IBM solidDB Linked Library Access User Guide*.

When solidDB is started, it checks if a database already exists. The server first looks for a `solid.ini` configuration file and reads the value of FileSpec parameter. Then the server checks if there is a database file with the names and paths specified in the FileSpec parameter. If a database file is found, then the solidDB will automatically open that database. If no database is found, then the server creates a new database.

Table 3. Starting the server

Operating System	**To Start the Server...**
UNIX/Linux	Enter the command solid at the command prompt. When you start the server for the first time, enter the command solid -f at the command prompt to force the server to run in the foreground.
Microsoft Windows	Click the shortcut, in the Start menu, labeled solidDB Server. Alternatively, enter the command solid at the command prompt in the server's working directory (by default, bin\, in the installation directory. To start the server to run in the background, enter the command start solid.
Open VMS	Enter the command run solid at the command prompt.

For details on the FileSpec parameter, read “FileSpec_[1...n] parameter” on page 50.

Creating a new database

If a database does not exist, solidDB will at start up automatically create a new database. In the Microsoft Windows environment, creating the database begins with a dialog prompting for the database administrator's username, password, and a name for the default database catalog. For details, read "Managing Database Objects" in *solidDB SQL Guide*.

In other environments, if you do not have an existing database, the following message appears:

```
Database does not exist. Do you want to create a new database (y/n)?
```

By answering "yes", solidDB prompts you for the database administrator's username, password, and a name for the default database catalog.

The username requires at least two characters. The maximum number of characters is 80. A user name must begin with a letter or an underscore.

The password requires at least three characters. The maximum number of characters is 80. Passwords can begin with any letter, underscore, or number. Use lower case letters from a to z, upper case letters from A to Z, the underscore character "_", and numbers from 0 to 9.

You cannot use the double quote (") character in the password. The use of apostrophe ('), semicolon (;), or especially space (' ') is strongly discouraged, because some tools may not accept these characters in the password.

Lowercase characters are converted to uppercase.

The catalog requires at least one character. The maximum number of characters is 39.

See also “Entering password from a file” on page 59.

Note:

If you plan to use solcon, do not create passwords with non-ASCII characters, because solcon does not perform UTF-8 translation for any input.

CAUTION:
The catalog name must not contain spaces.

Note: You must remember your username and password to be able to connect to solidDB. There are no default usernames ; the administrator username you enter when creating the database is the only username available for connecting to the new database.

After accepting the database administrator's username and password, solidDB creates the new database.

By default the database will be created as one file (`solid.db`) in the solidDB working directory. An empty database containing only the system tables and views uses approximately four megabytes of disk space. The time it takes to create the database depends on the hardware platform you are using. If you have a very

small database (less than four megabytes) and want to keep the disk space less than four megabytes, set the value of the ExtendIncrement parameter in the `solid.ini` configuration file to less than 500 (default). This parameter and other parameters are discussed in Appendix A, "Server-side configuration parameters," on page 117.

After the database has been created, solidDB starts listening to the network for client connection requests. In the Microsoft Windows environment, a solidDB icon appears, but in most environments solidDB runs invisibly in the background as a daemon process.

Login

solidDB database requires users to login to the database with their username and password.

If you try to login four times with an incorrect username and/or password, the system will block your IP address for a maximum of 60 seconds. This feature cannot be configured or switched off.

About solidDB databases

This section describes solidDB database structure and ways you can specify different values when creating solidDB databases.

solidDB configuration file (`solid.ini`)

When you start solidDB, it reads configuration parameters from the `solid.ini` configuration file.

The `solid.ini` file specifies parameters that help customize and optimize the solidDB database server. For example, the FileSpec parameter in the `solid.ini` file specifies the directory and file names of the data files in which the server stores the user data. Another parameter specifies the block size for the database. The block size affects performance and also limits the maximum record size. The FileSpec and BlockSize parameters are described in the next section.

You can find a complete description of all parameters, details about the proper format of the `solid.ini` file, and instructions for specifying `solid.ini` configuration parameters in Appendix A, "Server-side configuration parameters," on page 117. For more details about setting parameters, read 3, "Configuring solidDB," on page 47.

Setting up database environment

By default the solidDB database files, log, message, and trace files are created in the solidDB working directory. For production environments, you may want to set up an environment where, for example, database files, backup files, and log files are located on different disks.

Default working directory settings

A *working directory* is the directory that contains the files related to running a particular solidDB instance.

The following table shows the most common solidDB files, their factory value locations, and how to modify the locations.

Table 4. solidDB default files

File	Factory value location	How to modify
license file (`solid.lic`, `soliduc.lic`, or `solideval.lic`)	working directory	Define path in SOLIDDIR environmental variable
`solid.ini` configuration file	working directory	Define path in SOLIDDIR environmental variable
database files (`solid.db`)	working directory	Define with **IndexFile.FileSpec** parameter
transaction log files (`sol#####.log`)	working directory	Define location with **Logging.LogDir** parameter or Define location and file name with **Logging.FileNameTemplate** parameter **Note:** If you specify a directory for the log files, the directory must exist before you start solidDB: solidDB cannot create directories.
message file (`solmsg.out`)	working directory	Location and name cannot be changed; the `solmsg.out` file is always output in the working directory.
error file (`solerror.out`)	working directory	Location and name cannot be changed; the `solerror.out` file is always output in the working directory.
trace file (`soltrace.out`)	working directory	Define with **Com.TraceFile** parameter
backup files	<working directory>/ backup	Define with **General.BackupDirectory** parameter **Note:** The directory for the backup files must exist before you make a backupsolidDB: solidDB cannot create directories.

Recommendations for production environments

- If you do not wish to run the installer on your production environment node, install solidDB on a separate node and copy the executables, libraries, and drivers manually to your production node, as applicable for your setup.
- To prevent loss of data in case of a disk failure, store the database files and transaction log files on different physical drives. This will also provide best performance, especially during database checkpoints when both database files and transaction log files are written at the same time.
- Use local disks (instead of network disks) for storing the database files and log files.

  This is especially important with a solidDB HotStandby setup. The HotStandby configurations are targeted for environments with shared nothing architecture, and this is best achieved by having the primary and secondary databases in separate nodes, each using local disks. Network disks have a risk of being a logical/physical single point of failure in the system.

Related topics

- "Configuration files and parameter settings" on page 47
- "Managing database files and caching (IndexFile section)" on page 50

- "Viewing error messages and log files" on page 16
- "Performing backup and recovery" on page 29
- "Running several servers on one computer" on page 41

Setting database block size (BlockSize) and location (FileSpec)

The default block size for the solidDB database file is 16 KB. The block size is defined in multiples of 2 KB. The minimum block size is 2 KB and the maximum is 64 KB. The maximum size of the database is 64 TB.

The block size is set with the parameter **Indexfile.BlockSize**. If you want solidDB to create a database with a different block size, you have to set a new constant value before creating a new database. If you have an existing database, be sure to move the old database (.db) and log files (.log) to another directory; the next time you start solidDB, a new database is created.

To modify the constant value for the new database, add the following lines in the `solid.ini` file, providing the size in bytes :

```
[Indexfile]
BlockSize=size_in_bytes
```

The unit of size is 1 byte (as in all size-related parameters). The unit symbols of K and M (for KB and MB, respectively) can also be used.

After you save the file and start solidDB, it creates a new database with the new constant values from the `solid.ini` file.

Similarly, you can also modify the **FileSpec** parameter to define the following:

- location of the database file (the default is `solid.db` in the solidDB directory)
- maximum size (in bytes) the database file can reach (the default value is 2147483647, which equals 2 G-1 bytes). The maximum file size is (4 G-1)*blocksize. With the default 16 KB block size, this makes 64 TB - 1.

You can also use the **FileSpec** parameter to divide the database file into multiple files and onto multiple disks. This may be required if you want to create a large physical database.

For details on configuration with the **FileSpec** parameter, read "Managing database files and caching (IndexFile section)" on page 50.

Defining database objects

solidDB database objects include catalogs, schemas, tables, views, indexes, stored procedures, triggers, and sequences. By default, database object names are qualified with the object owner's user id and a system catalog name that you specify when creating a database for the first time or converting an old database to a new format. You can also specify that database objects be qualified by a schema name. For details, see section *Managing database objects* in the *IBM solidDB SQL Guide*.

solidDB supports a practically unlimited number of tables, rows, and indexes. Character strings and binary data are stored in variable length format. This feature saves disk space. It also makes programming easier on developers since the lengths of strings or binary fields do not have to be fixed. The maximum size for a single column value is 2G-1 bytes.

By configuring the **MaxBlobExpressionSize** parameter, you can set the maximum size of LONG VARCHAR (or CLOB) columns that are used in string functions. (The size can be specified in kilobytes (K) or megabytes (M).) By default, the size is 1MB (1 megabyte).

For efficiency, solidDB can store BLOB data outside the table. When BLOBs (Binary Large Objects), such as objects, images, video, graphics, or digitized sound are larger than a particular size, solidDB automatically detects this and stores the objects to a special file area that has optimized block sizes for large files. No administrative action is required. For more information, see section *BLOBs and CLOBs* in the *Appendix: Data Types* in the *IBM solidDB SQL Guide*.

Connecting to solidDB

Note: This section applies to standard solidDB only. If you are using solidDB with linked library access, refer to the corresponding section in *IBM solidDB Linked Library Access User Guide*.

After starting solidDB, you can test the configuration by connecting to the server from your workstation using the solidDB teletype tools, SQL Editor or Remote Control. Read 4, “Using solidDB data management tools,” on page 59, for details on these utilities, which are part of the solidDB Data Management tools.

To connect to solidDB:

1. View the `solmsg.out` file in your database directory for valid network names that you can use to connect to solidDB.

   The following messages indicate what names you can use.

   ```
   Listening of 'tcp hobbes 1313' started.
   ```

2. Start one of the following applications and give the network name of the server as a command line parameter:

Table 5. Connecting to solidDB

Tool	Command

solidDB Remote Control (solcon)	`solcon "networkname" [userid [password]]`
	For example:
	`solcon "tcp hobbes 1313"`
	If you did not specify the database administrator's user name and password on the command line, then solcon will prompt you to enter them.

solidDB SQL Editor (solsql)	`solsql "networkname" [userid [password]]`
	For example:
	`solsql "tcp hobbes 1313"`
	If you did not specify the database administrator's user name and password on the command line, then solsql will prompt you to enter them.

After a while you will see a message indicating that a connection to the server has been established.

Viewing error messages and log files

By default, solidDB outputs errors and messages in the `solmsg.out` and `solerror.out` log files in the solidDB working directory. To view the descriptions of single or all error messages, use `ADMIN COMMAND 'errorcode'`.

Controlling message log output

If you want to process the message files programmatically, you can enable the messages to be output with an 8-character unique code. You can also disable the generation of message log files.

solidDB maintains the following message log files:

- `solmsg.out` – log file for normal informational events, such as connects, disconnects, checkpoints, backups, failed logins and so on
- `solerror.out` – log file for fatal errors, typically causing the server to crash

Additionally, solidDB can also produce trace files (`soltrace.out`) for troubleshooting purposes.

You can view the message log files with a text editor.

The message log file size is controlled with the **Srv.MessageLogSize** parameter. When the maximum size of the message log file is reached, the current `solxxx.out` file is renamed to `solxxx.bak`, and a new `solxxx.out` file is started. To avoid overwriting the contents of the backup `solxxx.bak` message log the next time the maximum size of the message log file is reached, use the **Srv.KeepAllOutFiles** parameter to enable the log files to be named incrementally.

Enabling message codes in message logs

Each error and status message is identified with an 8-character unique code. If the message files are processed programmatically, it is easier to parse them if the message codes are included. To enable the message code output, set the **Srv.PrintMsgCode** to 'yes' (default is 'no').

Disabling message log generation

To disable the generation of the `solmsg.out` and the `solerror.out` log files, set the **Srv.DisableOutput** parameter to 'yes' (default is 'no').

Important: Disabling the generation of log files makes it difficult to diagnose problems. Turning off message logging will increase performance and reduce disk space usage; however, in most cases the improvement is minimal. This option is useful only in unusual situations, such as when I/O is "expensive" (as it is in some systems that use FLASH memory), or in systems where data storage space is extremely limited and the message log file accumulates indefinitely without being deleted.

Viewing error message descriptions with `ADMIN COMMAND 'errorcode'`

Each error and status message is identified with a unique number that you can use with `ADMIN COMMAND 'errorcode'` to view the error description.

The command ADMIN COMMAND 'errorcode <error_number>' displays the description of the given error message.

For example:

```
ADMIN COMMAND 'errorcode 14706';
       RC TEXT
       -- ----
        0 Code:  SRV_ERR_HSBINVALIDREADTHREADMODE (14706)
        0 Class: Server
        0 Type:  Error
        0 Text:  Invalid read thread mode for HotStandby, only mode 2 is supported.
4 rows fetched.
```

The command ADMIN COMMAND 'errorcode all' displays the descriptions of all error messages in a Comma Separate Value (CSV) format.

The error codes and their descriptions are also available in Appendix D, "Error codes," on page 179.

Using trace files

The trace files (soltrace.out) are needed primarily for troubleshooting of exceptional events.

Monitoring the trace files is not necessary for everyday operation of the server. For more details about the trace files and how to use them, see 7, "Diagnostics and troubleshooting," on page 111.

Tracing failed login attempts

When login fails, the information about the attempt is recorded for security reasons.

Failed attempt always

- raises a SYS_EVENT_ILL_LOGIN event, and
- prints message to both solmsg.out and solerror.out.

Messages include the IP address and the username of the attempt, for instance. The syntax of the message is as follows:

```
timestamp [message code] User username tried to
connect from {hostname | unnamed host} with an
illegal username or password. [SOLAPPINFO is solappinfo value.]
```

Example:

```
Thu May 12 17:55:17 2005
12.05 17:55:17 User 'FOO' tried to connect
from localhost.localdomain (127.0.0.1)
with an illegal username or password.
```

Note: The *message code* is only included if message code printing is enabled (**Srv.PrintMsgCode=yes**) in solid.ini.

Note: The SOLAPPINFO part is only included if the corresponding environment variable is set at the client computer.

Monitoring solidDB

The following sections describe the methods used for querying the status of a solidDB database.

Checking overall database status

The general server status may be retrieved by using the following command in solidDB SQL Editor (solsql):

```
admin command 'status';
       RC TEXT
       -- ----
        0 IBM solidDB started at 2008-12-04 12:48:24
        0 Current directory is C:\solidsw\solid63\eval_kit\standalone
        0 Using configuration file C:\solidsw\solid63\eval_kit\standalone\solid.ini
        0 Memory statistics:
        0     39269 kilobytes
        0 Process size statistics:
        0     Resident set size: 16312 kilobytes
        0     Virtual size: 43040 kilobytes
        0 Transaction count statistics:
        0     Commit Abort Rollback    Total Read-only  Trxbuf  Active Validate
        0        114     0        1      115       237       0       1        0
        0 Cache count statistics:
        0     Hit rate      Find      Read     Write
        0     100.0         28809     0         56
        0 Database statistics:
        0     Index writes         3623  After last merge      0
        0     Log writes           2277  After last cp         0
        0     Active searches         0  Average               1
        0     Database size       8064 kilobytes
        0     Log size              16 kilobytes
        0 User count statistics:
        0     Current Maximum Total
        0           1       1     1
```

The result set fields are described below:

- Memory statistics show the amount of memory solidDB has allocated from the operating system. This number does not include the size of the executable itself.
- Transaction count statistics show the number of different transaction operations since startup.
- Cache count statistics show cache hit rate and number of cache operations since startup. Cache hit rate usually should be above 95 percent. If it is below 95 percent, consider increasing the cache size.
- Database statistics show a number of the most important database operations since startup. The `Index writes - After last merge` is an important figure here. It reveals the size of the multi-versioning storage tree ofsolidDB, known as the "Bonsai Tree." The smaller this value is, the better the server performance. A large value indicates that there is a long-running transaction active in the engine. Note that an excessively large Bonsai Tree causes performance degradation. For details on reducing Bonsai tree size, read “Reducing Bonsai Tree size by committing transactions” on page 94.
- User count statistics shows the current and the maximum number of concurrent users.

Obtaining currently connected users

You can also obtain a listing of connected users by entering the following command in solidDB SQL Editor (solsql):

ADMIN COMMAND 'userlist';

The command provides the following kind of result set:

```
RC TEXT
-- ----
0 User name:     User id: Type:  Machine id:      Login time:
0 DBA   1        SQL    Local   27.05 16:13:22
```

Throwing out a connected solidDB user

To disconnect a single user from the server, enter the following command in solidDB SQL Editor (solsql):

```
ADMIN COMMAND 'throwout user_id';
```

Note that this command throws out user connections; it does not break the connection between a HotStandby Primary and HotStandby Secondary server.

Querying the status of the most recent backup

To obtain a status of the most recently run local backup, enter the following command in solsql:

ADMIN COMMAND 'status backup';

Obtaining the status of the most recently made network backup, enter the command:

ADMIN COMMAND 'status netbackup"

If the last backup is successful, the result set looks as follows:

```
RC TEXT
-- ----
0 SUCCESS
```

If the latest backup has failed, then the RC column returns an error code. Return code 14003 with text "ACTIVE" means that the backup is currently running.

Producing a status report

To create a report about the current status of solidDB, enter the following command in solidDB SQL Editor (solsql):

```
ADMIN COMMAND 'report report_filename'
```

This report is primarily meant for solidDB internal use only because it contains information that requires very detailed understanding about the internals of solidDB. End users sometimes are requested to produce the report for troubleshooting purposes.

Performance counters (perfmon)

You can get information about various database operations and performance with `ADMIN COMMAND 'perfmon'`.

The `ADMIN COMMAND 'perfmon'` command returns a result set of all solidDB performance counters (called *perfmons* or *pmons*). All the counters are listed and described in "Full list of perfmon counters" on page 21.

```
ADMIN COMMAND 'perfmon';
       RC TEXT
       -- ----
        0 Performance statistics:
        0 Time (sec)                              3    Total
        0 File open                        :    0.0      0.1
        0 File read                        :    0.0      1.2
        0 File write                       :    0.0      0.0
        0 File append                      :    0.0      0.6
        0 File flush                       :    0.0      0.0
        0 File lock                        :    0.0      0.0
        0 Cache find                       :    0.0     78.5
        0 Cache read                       :    0.0      1.0
        0 Cache write                      :    0.0      0.0
        0 Cache prefetch                   :    0.0      0.0
        0 Cache prefetch wait              :    0.0      0.0
        0 Cache preflush                   :    0.0      0.0
        0 Cache LRU write                  :    0.0      0.0
        ...
```

Each column represents a snapshot of the performance information that reflects the most recent few minutes. The first column shows average performance information from a period of seconds. The `Total` column shows average information since solidDB was started.

Most numbers are events/second. Those numbers that cannot be expressed as events/second (for example, database size) are expressed in absolute values.

The command syntax also has options that allow you to specify output options. For details on these options, see the perfmon option in Appendix E, "solidDB ADMIN COMMAND syntax," on page 257.

You can restrict the output by providing a list of prefixes of counter names. For example, `ADMIN COMMAND 'perfmon db'` returns all pmon counters starting with 'db':

```
ADMIN COMMAND 'perfmon db';
RC TEXT
 -- ----
 0 Performance statistics:
 0 Time (sec)                              42     43     43     42     30   42     43     26    Total
 0 DBE insert                       :    0.0    0.0    0.0    0.0    0.0  0.0    0.0    0.0      0.0
 0 DBE delete                       :    0.0    0.0    0.0    0.0    0.0  0.0    0.0    0.0      0.0
 0 DBE update                       :    0.0    0.0    0.0    0.0    0.0  0.0    0.0    0.0      0.0
 0 DBE fetch                        :    0.0    0.0    0.0    0.0    0.0  0.0    0.0    0.0      0.7
 0 DBE dd operation                 :      0      0      0      0      0    0      0      0        0
 0 Db size                          :   8064   8064   8064   8064   8064 8064   8064   8064     8064
 0 Db free size                     :   7472   7472   7472   7472   7472 7472   7472   7472     7472
 0 DB actiongate lock time, latest:      0      0      0      0      0    0      0      0        0
 0 DB actiongate lock time, sum     :      0      0      0      0      0    0      0      0        0
 0 DB actiongate lock count         :      0      0      0      0      0   0      0      0        0
12 rows fetched.
```

Producing a continuous performance monitoring report

The command `ADMIN COMMAND 'perfmon diff'` allows you to start and stop producing continuous performance counter reports to a file.

To start monitoring:

```
ADMIN COMMAND 'perfmon diff start filename interval'
```

For example, to start logging all counters, with 1 second interval:

```
ADMIN COMMAND 'pmon diff start counter_log.csv 1000'
```

This will log the counter data to a comma-separated values (CSV) file starting with a row of counter names, and having one row per each sampling time.

To stop monitoring:

```
ADMIN COMMAND 'pmon diff stop'
```

Full list of perfmon counters

The counters are listed in the order they appear in the output report.

Table 6. Perfmon counters

Perfmon Variable	Description
Time (sec)	In one-time report: length of the measurement time interval, in seconds. The latest interval is on the right side of the table.
TimeMs	In a differential report: measurement time interval, in milliseconds. The oldest interval is in the first row of the table.
File open	File open calls/sec
File read	File read calls/sec
File write	File write calls/sec
File append	File append calls/sec
File flush	File flush calls/sec
File lock	File lock calls/sec
Cache find	Cache fetches/sec
Cache read	Cache misses/sec
Cache write	Cache page flushes/sec
Cache prefetch	Cache prefetched pages/sec
Cache prefetch wait	Cache waits for prefetched pages/sec
Cache preflush	Preflushing cache pages/sec
Cache LRU write	A write from cache is done when performing an LRU replacement. This indicates that the client thread must write one block to disk before reading a new block from the disk because there has not been a free disk block available. A very high value can indicate just high I/O load, or it can indicate that I/O preflusher values are not optimal.
Cache slot wait	This counter indicates that there is concurrent access to the same block and one thread must wait for the other. Depending on the cache configuration, it can also indicate that the mutex count for the cache is not optimal and there are false conflicts. The default mutex count does not cause false conflicts here.
Cache slot replace	Database cache slot is replaced and old slot is removed.
Cache write storage leaf	Database cache has written a storage tree leaf page to disk.
Cache write storage index	Database cache has written a storage tree index page to disk.
Cache write bonsai leaf	Database cache has written a Bonsai-tree leaf page to disk.
Cache write bonsai index	Database cache has written a Bonsai-tree index page to disk.

Table 6. Perfmon counters (continued)

Perfmon Variable	Description
RPC messages	Total number of sent messages/sec
RPC read	Total number of read messages/s
RPC write	Total number of write messages/sec
RPC uncompressed	When RPC compression enabled, number of bytes/sec
RPC compressed	When RPC compression enabled, number of compressed byte/s
Com sel empty	TCP socket select nil returns/sec
Com sel found	TCP socket select successes/sec
SQL prepare	SQL prepare statements/sec
SQL execute	SQL execute statements/sec
SQL fetch	SQL fetch statements/sec
DBE insert	Table engine row inserts/sec
DBE delete	Table engine row deletes /sec
DBE update	Table engine row updates /sec
DBE fetch	Table engine row fetches /sec
DBE dd operation	Server has executed SQL data dictionary operation.
Proc exec	Procedure executions/sec
Trig exec	Trigger executions/sec
SA insert	SA-level row inserts/sec
SA delete	SA-level row deletes/sec
SA update	SA-level row updates/sec
SA fetch	SA-level row fetches/sec
Trans commit	Committed transactions/sec
Trans abort	Aborted transactions/sec
Trans rollback	Rolled back transactions/sec
Trans readonly	Read-only transactions/sec
Trans buf	Current® transaction buffer size
Trans buf cleanup	Cumulative number of cleanup operations since startup

Table 6. Perfmon counters (continued)

Perfmon Variable	Description
Trans buf added	Cumulative number of transactions added since startup
Trans buf removed	Cumulative number of transactions removed since startup
Trans validate	Current number of active commit-time validations
Trans active	Current number of active transactions
Trans read level	This counter indicates the current transaction read level. This counter value increases all the time. Because the counter value is 32-bit variable, it can have a negative value, but still logically the value is increasing. If the value stays the same for a long time with concurrent write transactions, it indicates that a long transaction is blocking the read level and can cause merge blocking and an increase in the Bonsai tree size.
Ind write	Index writes/sec
Ind nomrg write	number of nonmerged rows (committed and uncommitted)
Log write	Log record writes/sec
Log file write	Log block writes/sec
Log nocp write	Pending log records since last checkpoint
Log size	Total size of log file, in KB
Search active	Table engine-level active searches.
Db size	Total database size on disk, in KB
Db free size	Free space in the database (page level), in KB
Mem size	Total size of dynamically allocated memory, in KB
Merge quickstep	Quick merge steps/sec
Merge step	Full merge steps/sec
Merge step (purge)	Node split-inflicted merge keys/sec (if enabled)
Merge step (user)	User thread-activated merge row/sec
Merge oper	Lower-level merge operations/sec
Merge cleanup	Transaction buffer cleanup calls/sec (if split purge enabled)
Merge active	Yes/no (1/0)
Merge nomrg write	Current number of index entries waiting for merge
Merge file write	Merge-inflicted file writes/sec

Table 6. Perfmon counters (continued)

Perfmon Variable	Description
Merge file read	Merge-inflicted file reads/sec
Merge level	Current merge level (read level of the oldest active transaction)
Backup step	Database backup steps/sec (also in netbackup and netcopy)
Backup active	Yes/no (1/0)
Checkpoint active	Yes/no (1/0)
Checkpoint count	Checkpoint serial no. from startup
Checkpoint file write	Checkpoint file writes/sec
Checkpoint file read	Checkpoint file reads/sec
Est read samples	Estimator sample refresh call/s
Sync repl msg forw	Replica: fowarded messages/sec
Sync repl msg getr	Replica: received message replies/sec
Sync repl msg exec	Replica: executed messages/sec
Sync mast msg read	Master: message reads/sec
Sync mast msg exec	Master: message execs/sec
Sync mast msg write	Master: message writes/sec
Sync mast subs	Master: refreshes/sec
Log flush (L)	Logical log flushes/sec (e.g. commit)
Log flush (P)	Physical log flushes/sec
Log grpcommwkup	Group commit wakeups/sec
Log flush full	Log page full flushes/sec
Log wait flush	Current number of user threads waiting for log operation
Log writeq full rec	Log writes while log write queue full (in number of records)
Log writeq records	Number of records in current log writer queue.
Log writeq bytes	Number of bytes in log writer queue.
Log writeq pending bytes	Number of bytes for the next log writer queue flush.
Log writeq add	Number of records added to log writer queue.
Log writeq write	Number of records written from log writer queue to log file.
Log writeq full byt (byte size)	Log writes while log write queue full (in bytes)
HSB operation count	Primary/Secondary: transferred log record/sec

Table 6. Perfmon counters (continued)

Perfmon Variable	Description
HSB commit count	Primary: commit record/sec
HSB packet count	Primary: messages/sec
HSB flush count	Primary/Secondary: message flushes/sec
HSB cached bytes	Primary/Secondary: current size memory based log buffer, in bytes
HSB cached ops	Primary/Secondary: current size of the memory-based log buffer, in operations (log records)
HSB flusher bytes	Number of bytes of the HSB log in the send queue to the Secondary
HSB notsent bytes	Number of bytes in the HSB log that has been accumulated (for example, during a catchup) and not sent to the Secondary yet
HSB grouped acks	Secondary: current number of ack groups (physical acks)
HSB state	Name of the current HSB state
HSB wait cpmes	Yes/no (1/0) Primary: waiting for checkpoint ack from the Secondary
HSB secondary queues	Secondary: current number of queues pending processing
HSB log reqcount	HSB log write requests/sec
HSB log waitct	HSB log waits-for-write requests/sec
HSB log freespc	HSB: number of log operations there is space for in the protocol window
HSB catchup reqcnt	HSB log write requests/sec, for catchup
HSB catchup waitcnt	HSB log waits-for-write requests/sec, for catchup
HSB catchup freespc	HSB: number of log operations there is space for in the protocol window, for catchup
HSB alone freespc	Primary: in Primary alone, bytes there is room for in the transaction log
Thread count	Current number of threads
Trans wait readlvl	Waits/sec for read level at commit
Lock ok	Successful lock requests/sec
Lock timeout	Lock timeouts/sec
Lock deadlock	Deadlocks/s
Lock deadlock check	Number of lock manager deadlock checks done.

Table 6. Perfmon counters (continued)

Perfmon Variable	Description
Lock deadlock loop	Number of lock manager deadlock check loops done.
Lock wait	Lock waits/sec
Lock count	Number of locks in lock manager.
Dropped search buffers	Number of search buffers removed from disk based table searches because too many buffers were used.
Number of search buffers	Current number of search buffers used for disk based tables.
NOCHECK operations	Internal number of nocheck operations performed.
MME cur num of locks	Current no. IME locks
MME max num of locks	Peak number of IME locks (since startup)
MME cur num of lock chains	Current no. IME hash buckets
MME max num of lock chains	Peak no. IME hash buckets (since startup)
MME longest lock chain path	IME: longest hash overflow path
MME mem used by tuples	IME memory allocated to tuples in kilobytes
MME mem used by indexes	IME memory allocated to indexes in kilobytes
MME mem used by page structs	IME memory allocated to the shadow structures in kilobytes
MME inserts with x gate	Number of inserts done in exclusive mode. Insert switches from shared mode to exclusive mode for example, when the insert causes index node split.
Posted events queue	Number of posted events that has not been consummated by the subscribers
Index search both	Search is done from both the Bonsai tree and the storage tree
Index search storage	Index search is done from storage tree only
B-tree node search keys	DBE B tree searches/sec
B-tree node search mismatch	A search was done by using the mismatch index search structure within a B-tree node. Mismatch index is a search structure where anarray of mismatch index positions is built within a B-tree node. This mismatch index is a compact and linear data structure that is used to perform a fast scan over compressed key information to find a key position within the B-tree node. It attempts to optimize the search by using fast access in the processor cache row by packing relevant search information in one to three processor cache pages.
B-tree node build mismatch	A new mismatch index search search structure is built within a B-tree node. Mismatch index is a search structure where an array of mismatch index positions is built within a B-tree node. This mismatch index is a compact and linear data structure that is used to perform a fast scan over compressed key information to find a key position within the B-tree node. It attempts to optimize the search by using fast access in the processor cache row by packing relevant search information in one to three processor cache pages.

Table 6. Perfmon counters (continued)

Perfmon Variable	Description
B-tree node split	DBE B tree node splits/sec
B-tree node relocate	A B-tree node is relocated. This happens when a block that belongs to a previous checkpoint is changed for the first time. Typically, this value is highest immediately after a checkpoint.
B-tree node delete empty	An empty B-tree node is deleted.
B-tree node exclusive	Exclusive access to the B-tree is used. This can happen, for example, in a node split case such as when the tree root is split.
B-tree key read	Normal key value is read from the B-tree.
B-tree key read delete	Delete mark is read from the B-tree.
B-tree key read oldversion	Old row version is read from the B-tree.
B-tree key read abort	A row from an aborted transaction is read from the B-tree. This includes all transactions that were not successfully completed.
B-tree storage leaf len	Average length for storage tree leaf node.
B-tree storage index len	Average length for storage tree index node.
B-tree bonsai leaf len	Average length for Bonsai-tree leaf node.
B-tree bonsai index len	Average length for Bonsai-tree index node.
Bonsai-tree height	Current Bonsai tree height in levels.
B-tree lock node	Number of B-tree node lock calls.
B-tree lock tree	Number of whole B-tree lock calls.
B-tree lock full path	Number of B-tree full node path lock calls.
B-tree lock partial path	Number of B-tree partial node path lock calls.
B-tree get no lock	Number of B-tree no lock calls.
B-tree get shared lock	Number of B-tree shared lock calls.
Pessimistic gate wait	Number of waits for pessimistic disk based table gate.
Merge gate wait	Number of waits for merge gate.
Storage gate wait	Number of waits for storage tree gate.
Bonsai Gate wait	Number of waits for Bonsai-tree gate.
Gate wait	There is a wait in a gate object. A gate object is an internal synchronization mechanism.
Logreader spm reqcount	Logreader log space request/sec
Logreader spm waitct	Logreader log space waits/sec
Logreader spm freespc	Logreader: number of log operations the protocol window has space for.
Logreader logdata queue len	Logreader: number of log record blocks waiting for processing.
Logreader record queue len	Logreader: number of log records waiting for propagation.
Logreader stmt queue len	Logreader: number of statements waiting for statement commit/rollback.

Table 6. Perfmon counters (continued)

Perfmon Variable	Description
Logreader open cursors	Logreader: number of open cursors to SYS_LOG.
Logreader records processed	Logreader: number of log records processed/sec.
Logreader records sent	Logreader: number of log records sent for propagation/sec.
Logreader commits processed	Logreader: number of commits processed/sec.
Logreader commits sent	Logreader: number of commits sent to the propagator/sec.
Logreader messages sent	Logreader: number of wakeup messages to open cursors/sec.
Logreader catchup state	Logreader catchup state.
Logreader catchup queue len	Logreader: number of log records in catchup queue.
Logreader catchup queue size	Logreader: size of the catchup queue, in bytes.
Logreader pending queue len	Logreader: number of pending log records in the in-memory log buffer.
Logreader memcache queue len	Logreader: length of the in-memory buffer queue, in operations.
Logreader batch queue len	Logreader: current number of operations queued for the next batch.
Logreader flush batch full	Logreader: a full transaction back was flushed from logreader.
Logreader flush batch force	Logreader: a non-full transaction batch was flushed from logreader.
TS applied transactions	Number of transactions applied into solidDB by CDC instance when solidDB is a target datastore.
DB actiongate lock time, latest	Amount of time in milliseconds the last lock lasted
DB actiongate lock time, sum	Amount of time in milliseconds all locks have lasted since server startup
DB actiongate lock count	Number of locks since server startup

Shutting down solidDB

Note: This section applies to standard solidDB only. If you are using solidDB with linked library access, read the corresponding section in *IBM solidDB Linked Library Access User Guide*.

You can shut down the solidDB in the following ways:

- Programmatically from an application such as solidDB Remote Control, or solidDB SQL Editor. To do this, perform the steps below.

  Note: When using solidDB SQL Editor for steps 1-3 below, enter the full SQL Syntax,

  ```
  ADMIN COMMAND 'command_name'
  ```

  (for example, ADMIN COMMAND 'close')

1. To prevent new connections to solidDB, close the database(s) by entering the following command:

   close

   Note that you can revert the effect by entering the command:

   open
2. Exit all users of solidDB (except the current connection) by entering the following command:

   throwout all

   Note that this command does not wait for open transactions to finish; it aborts and rolls back all open transactions.
3. Stop solidDB by entering the following command:

   shutdown

- Using command ADMIN COMMAND 'shutdown force" that includes all of the above.
- Right-clicking the server icon and selecting **Close** from the menu appearing in the Microsoft Windows environment.
- Remotely, using the command 'net stop' through the Windows system services. Note that you may also start up solidDB remotely, using the 'net start' command.

Each of these shutdown mechanisms will start the same routine, which writes all buffered data to the database file, frees cache memory, and finally terminates the server program. Shutting down a server may take a while since the server must write all buffered data from main memory to the disk.

Performing backup and recovery

Backups are made to secure the information stored in your database files. If your database files have become corrupted or they are lost due to a system failure, you can restore the database from the backup files. To ensure that data is secure in the event of a system failure, you should regularly back up master and possibly also the replica databases.

solidDB main memory engine supports both local backups and backups made over the network, that is, network backups. Local backup produces a copy — one database file — of the current logical database, which possibly consists of multiple files. Network backup does the same except that the backup database is sent over the network to Network Backup Server.

This section describes how to back up your solidDB in-memory databases and recover from system failure. Furthermore, means of configuring, administering, and monitoring backup operations are presented. For guidelines for backing up and restoring the master and replica databases, see the *solidDB Advanced Replication Guide*.

Making local backups

You can initiate a local backup by entering the following command in solsql:

```
ADMIN COMMAND 'backup [-s] [dir backup dir]'
```

Available options for the backup command:

Table 7. Options for the backup command

Option	Description
-s	Synchronized execution. The call returns either when the backup is completed or due to an error.
dir	*backup dir* is a path expression determining the backup directory in the local file system. If the backup directory is omitted, it must be specified in the `solid.ini` configuration file. If the specified backup directory does not exist, solidDB database error 10030 is given. For more information on this error, see Appendix D, "Error codes," on page 179

The backup directory can be set beforehand in the configuration file by setting the parameter BackupDirectory in the [General] section of the configuration file. For the full list of available configuration parameters see Appendix A, "Server-side configuration parameters," on page 117.

CAUTION:
If two databases are copied to the same directory, the earlier will be overwritten by the latter. The *backup dir* must be different at least for each database. Moreover, although database files may be stored to different directories and partitions at the source server they all are copied to the same backup directory. Therefore equally named database files will conflict in the backup directory. As a consequence, only the last backed-up file among the equally named ones has backup copy in the backup directory.

Making backups over network

A network backup command may be sent to any host running a solidDB server. A server playing the role of the backup receiver is called a NetBackup Server.

Making netbackup

You can initiate a network backup ("netbackup" for short) by entering the following command in solsql:

```
ADMIN COMMAND 'netbackup [options] [DELETE_LOGS | KEEP_LOGS]
[connect connect str] [dir backup dir]'
```

Available options for the netbackup command:

Table 8. Options for the netbackup command

Option	Description
-s	Synchronized execution. The call returns either when the netbackup is completed or due to an error.
connect	*connect str* is an elementary connect string specifying the connection to NetBackup Server. If the connect string is omitted it must be specified in the `solid.ini` configuration file.

Table 8. Options for the netbackup command (continued)

Option	Description
dir	*backup dir* is a path expression determining the backup directory in NetBackup Server. The path can be either absolute or relative to the netbackup root directory. If the backup directory is omitted it must be specified in the `solid.ini` configuration file.
DELETE_LOGS	Delete backed-up log files in the source server. The backup using DELETE_LOGS is sometimes referred to as *Full backup*. This is the default value.
KEEP_LOGS	Keep backed-up log files in the source server. The backup using KEEP_LOGS is sometimes referred to as *Copy backup*. Using the keyword KEEP_LOGS corresponds to setting the General parameter NetbackupDeleteLog to "no".

For the full connect string syntax see "Format of the connect string" on page 49. For the full ADMIN COMMAND syntax see Appendix E, "solidDB ADMIN COMMAND syntax," on page 257.

CAUTION:
If two databases are copied to the same directory, the earlier will be overwritten by the latter. The *backup dir* should never point, for instance, to the root directory of the Netbackup Server.

Note:

- The command `ADMIN COMMAND 'netbackup'` is not supported within the **Srv.At** configuration parameter.
- The `ADMIN COMMAND 'status netbackup'` is a synonym of `ADMIN COMMAND 'status backup'` and reports on both local and network backups.
- The `ADMIN COMMAND 'netbackuplist'` is a synonym of `ADMIN COMMAND 'backuplist'` and reports on both local and network backups.

Flat and deep NetBackup directory structures

The NetBackup Server sees all the database files sent to it as one logical database even though the source database may consist of multiple files stored in different directories and on different permanent storage devices. By default, netbackup copies all the files of the source database to a single directory, that is, the user-specified netbackup directory.

It is, however, possible to explicitly specify the directories, the names and sizes of the backup files stored into the file system of the NetBackup Server. This is done by creating a `backup.ini` netbackup configuration file to the netbackup directory. The netbackup configuration file follows the syntax of `[IndexFile]` section in solidDB configuration file. Therefore, in addition to the section name, it may include multiple specifications for file names and sizes. Formally the syntax is as follows:

```
[IndexFile]
FileSpec_[1...N]=[path/]file name [maximum file size]
```

A NetBackup Server having such a `backup.ini` file receives the incoming database as a whole, splits it into N separate parts and stores the parts as files in accordance with the specifications in the `backup.ini` file.

Tip:

An easy way to retain the directory structure of the source server is to copy and rename the source server's `solid.ini` to `backup.ini` and move it to the backup directory at the NetBackup Server. The NetBackup Server reads only the FileSpec_[1...N] specifications from the [IndexFile] section, creates similar directory structure and stores backup files with their original properties to the NetBackup Server.

Configuring and automating backups

For both local and network backup, all the optional settings except the synchronized execution, -s, can be set beforehand in the database configuration file. Since the name and the syntax of the configuration parameters differ from the ADMIN COMMAND options, the corresponding parameter-option pairs are listed in the table below.

Corresponding ADMIN COMMAND options and configuration parameters for local backup

Table 9. Parameter correspondence to the `solid.ini` file for local backup

Option	**Value**	**parameter in section [General] of `solid.ini`**
dir	*backup dir*	BackupDirectory = *backup dir* default: no default

Corresponding ADMIN COMMAND options and configuration parameters for netbackup

Table 10. Parameter correspondence to the `solid.ini` file for netbackup

option	**value**	**parameter in section [General] of `solid.ini`**
connect	*connect str*	NetBackupConnect = *connect str* default: no default
dir	*backup dir*	NetBackupDirectory = *backup dir* default: no default
netbackup	DELETE_LOGS	NetbackupDeleteLog = yes default: yes
netbackup	KEEP_LOGS	NetbackupDeleteLog = no default: yes

For the complete list of configuration parameters and ADMIN COMMAND options see Appendix A, “Server-side configuration parameters,” on page 117 and Appendix E, “solidDB ADMIN COMMAND syntax,” on page 257, respectively.

Note: The options entered in ADMIN COMMAND command override corresponding parameters specified in the `solid.ini` database configuration file.

Making backups can be automated by using timed commands. Read "Entering timed commands" on page 41 for details.

What happens during backup

Both local and network backup create a self-contained and self-consistent image of a database by copying necessary files to the user-specified backup directory.

Every backup makes a checkpoint as its first action. This guarantees that the possible restore starts with as fresh backup as possible. This way, the slower roll-forward portion of the restore is minimized. The following files are then copied by default to the specified backup directory:

- the database files containing the checkpointed database itself,
- the log files including changes made by those transactions that are active when the backup takes place,
- the `solmsg.out` database message file (this is for convenience in diagnosing problems — the message file is not required during a restore), and
- the `solid.ini` configuration file is also copied by default because after a disk crash the original might be destroyed (the configuration file is not required during a restore).

The `solid.lic` licence file is not automatically copied.

Note: The name of the database files and their maximum size are specified in the **FileSpec[1...N]** parameters in the `[IndexFile]` section of the `solid.ini` configuration file. The name and location of log files is specified in the `[Logging]` section of the configuration file.

The log files are typically deleted from the source server after they have been copied to the backup directory since they have become useless. This is the default backup procedure and it is referred to as *Full backup*.

It is, however, possible to retain all the log files produced over time by the update transactions in the database server directory. Keeping all the log files is space-consuming but allows, for instance, bringing the database up-to-date by re-executing all the updates by using the log files only. This backup type is called *Copy backup*.

Note: If you want to use Copy backups, that is, retain the full log file history, you also must ensure that the log files are not deleted at the end of checkpoint. This can be done by ensuring that you do not have the line **CheckpointDeleteLog=yes** in section `[General]` of the `solid.ini` configuration file.

Local backup

In local backup the database and the log files are copied from the database directory to user specified backup directory accessible from within the same machine.

If the backup directory already includes files with same names, they will be overwritten. If the specified backup directory does not exist, the backup fails and the call returns an error.

CAUTION:

Ensure that backup and database directories are both on different physical device and in different file system than database files. If one disk drive is damaged, you will lose either your database files or backup files but not both. Similarly, if one file system fails, either the backup or the database files will survive.

Network backup

Netbackup is a facility for storing the whole database at some remote location. This is done by way of a solidDB Netbackup Server whose function is to receive backups over the network. One Netbackup Server can serve multiple simultaneous backup source servers.

Similarly to local backup, the files are written into a user specified directory in the Netbackup Server. If the target netbackup directory includes files with the same names, they will be overwritten. Unlike the local backup, if the specified remote directory does not exist, it is created automatically.

solidDB Netbackup Server requires the administrator privileges from the caller of netbackup. Less privileged users can perform netbackups by using stored procedures that are created by an administrator. In that case the user must be granted the right to execute the procedure.

Netbackup can be performed between different server versions provided that they are netbackup compatible. By principle, a newer version of the Netbackup Server will serve older versions of source servers. In other cases, the protocol version is checked and an incompatibility error is returned at the netbackup's request.

Administering network backup server

Every solidDB database server since version 4.5 also acts as a Network Backup Server. One configuration parameter, however, must be set in the in [Srv] section in the `solid.ini` configuration file:

`NetBackupRootDir=`*`netbackup root path`*

The path is relative to the working directory and the default is the working directory.

You can shut down a Netbackup Server by following the normal shutdown sequence and using the normal close and shutdown commands.

1. ADMIN COMMAND 'close'

   No new netbackup requests are accepted.
2. ADMIN COMMAND 'throwout all'

   Aborts the backups in progress.
3. ADMIN COMMAND 'shutdown"

   Shuts down the server.

Monitoring and controlling backups

solidDB offers a set of commands for monitoring and controlling backups. Backups can be controlled both by using the ADMIN COMMAND syntax in solsql.

Local backup and netbackup on source-server side

You can query and control backup processes by using the ADMIN COMMAND -SQL extension in solsql. The syntax is as follows:

```
ADMIN COMMAND 'command'
```

where the command may be any of those presented in the table below.

Table 11. Available backup and netbackup commands

Local Backup	Network Backup	Description
status backup	status netbackup	Displays the status of the most recent backup.
backuplist	netbackuplist	Displays a status list of last backups.
info bcktime		Displays the time of the latest completed backup.
abort backup	abort netbackup	Cancels the on-going backup process.

Query the list of all completed backups and their success status

To query the list of all completed backups and their success status, use the command:

ADMIN COMMAND 'backuplist'

Abort an active network backup operation

To abort an active network backup operation, use the command:

ADMIN COMMAND 'abort netbackup'

Correcting a failed backup

When solidDB is performing a backup — local or network — the command

```
ADMIN COMMAND 'status [backup | netbackup]'
```

returns the value "ACTIVE". The default option is backup. Once the backup is completed, the command returns either "OK" or "FAILED".

If the backup failed, you can find the error message that describes the reason for the failure in the `solmsg.out` file in the database directory. Correct the cause of the error and try again.

Typical problems in backups

Backup media is out of disk space. Making a backup requires the same amount of disk space as the database being backed-up. Therefore be sure you have enough disk space in the backup storage device.

Invalid path for backup directory. The backup directory you enter must be a valid path name in the server operating system. For example, if the server runs on a UNIX operating system, path separators must be slashes, not backslashes.

The local backup directory does not exist. Specifying a non-existent backup directory causes the server to print an error message and the backup fails. If you perform backups as timed operations you can ensure the success of backups from `solmsg.out` file.

The local backup directory is the same as that of the database. Since the backup copies database files with their original names to the target directory, using same source and target directories would lead to file sharing conflict.

solidDB network backup server does not exist in the specified location. Trying to start a network backup without setting up solidDB network backup server properly will fail the netbackup.

Restoring backups

You can restore the database to the state it was in when the backup was created by following the instructions below. Furthermore, you can revive a backup database to the current state by using log files generated after the backup was made. Those log files include information about the data inserted or updated since the latest backup.

Preparing netbackup files for recovery

Two preliminary steps may have to be taken before a database can be recovered from remote backup files.

1. If `backup.ini` was not used, the original naming and sizing of the database files must be restored from the `solid.db` file.
2. All the backup files must be copied to the node where the restore takes place.

Besides these steps, restoring a netbackup is similar to restoring local backup.

Returning to the state of the last backup

1. Shut down solidDB, if it is running.
2. Delete all log files from the log file directory. The default log file names are `sol00001.log`, `sol00002.log`, etc.
3. Copy the database files from the backup directory to the database file directory.
4. Start solidDB.

This method will not perform any recovery because no log files exist.

Refreshing database from the backup to the current state

1. Shut down solidDB, if it is running.
2. Copy the database files from the backup directory to the database directory.
3. Copy the log files from the backup directory to the log directory. If the same log files exist in both directories, do not overwrite the newer log files with the older backup log files.
4. Start solidDB.

solidDB will automatically use the log files to perform a roll-forward recovery.

Recovering from abnormal shutdown

If the server was closed abnormally, that is, if it was not shut down using the procedures described earlier, solidDB automatically uses the log files to perform a roll-forward recovery during the next start up. No administrative procedures are required to start the recovery.

Transaction logging

Transaction logging guarantees that no committed operations are lost in the case of a system failure. When an operation is executed in the server, the operation is also saved to a transaction log file. The log file is used for recovery in case the server is shut down abnormally.

There are two different logging modes:

- *Ping-pong method*

  This method uses the last two allocated disk blocks in the log file to write the two latest versions of the same logical incomplete disk block. The ping-pong method toggles between these two blocks until one block becomes full.

- *Overwriting method*

  This method rewrites in complete blocks at each commit until it becomes full. It may be used when data loss from the last log-file disk block is affordable.

solidDB allows you to decide whether you want to use logging or not. If logging is used, abnormally shut down databases can be restored to the state they were at the moment the failure took place. If the logging is disabled, databases can be restored to the backup state only. Transaction logging is enabled by default. If the full transaction recovery is not needed, logging can be disabled. To do this, set the [Logging] parameter **LogEnabled** to "no".

Logging may be synchronous or asynchronous, depending on the transaction durability setting. For more on transaction durability, see the subsection *Logging and transaction durability* in 5, “Performance tuning,” on page 83.

Creating checkpoints

A checkpoint updates the database file(s) on disk. Specifically, a checkpoint copies pages from the database server's memory cache to the database file on the disk drive. The server does the copy in a transactionally-consistent way; in other words, it only copies the results of committed transactions. The result is that all of the data in the database file is committed data from complete transactions. If the server fails between checkpoints, the disk drive will have a consistent and valid (although not necessarily up-to-date) snapshot of the data.

In between checkpoints, the server writes committed transactions to a transaction log. If the server fails, any transactions committed since the last checkpoint can be recovered from this transaction log. After a system crash, the database will start recovering transactions from the latest checkpoint.

Conceptually, you can think of checkpoints as being the main write operations to the database files on disk. The server does not write the results of each individual insert/update/delete statement (or even the result of each transaction) to the disk as it happens; instead the server accumulates committed transactions (in the form of updated pages in memory) and writes them to the disk only during checkpoints. (The server may also use part of the database file as swap space if the server's cache overflows. In this situation, the server will also write to the database file.)

Before and after a database operation, you may want to create a checkpoint manually. You can do this programmatically from your application with SQL command

ADMIN COMMAND 'makecp'

(Make CheckPoint). You can also force a checkpoint using a timed command. Read “Entering timed commands” on page 41 for details.

solidDB has an automatic checkpoint creation daemon, which creates a checkpoint after a certain number of writes to the log files. For more information about controlling the frequency of checkpoints, see “Tuning checkpoints” on page 93.

Checkpoints apply also to persistent in-memory tables, not just disk-based tables.

Note:

There can only be one checkpoint in the database at a time. When a new checkpoint is created successfully, the older checkpoint is automatically erased. If the server process is terminated in the middle of checkpoint creation, the previous checkpoint is used for recovery.

A checkpoint can require a substantial amount of I/O, and may affect the server's responsiveness while the checkpoint is occurring. For more details, read “Tuning checkpoints” on page 93.

Closing a database

You can close the database, which means no new connections to the database are allowed. To do this, issue the following command in solidDB SQL Editor (solsql):

ADMIN COMMAND 'close';

You use the close command when you want to prevent users from connecting to the database. For example, when you are shutting down solidDB, you must prevent new users from connecting to the database. As part of the shut down procedure you use the close command. Read “Shutting down solidDB” on page 28 for procedures to shut down a database.

After closing the database, connections from solidDB Remote Control will only be accepted. Closing the database does not affect existing user connections. When the database is closed no new connections are accepted (clients will get solidDB Error Message 14506).

To revert the effect of the close command, use:

ADMIN COMMAND 'open';

Running solidDB as a Windows service

solidDB can be run as a service in Windows. The first time you want to run solidDB as a service, you must install the service, that is, allow Windows to run solidDB as a service. After that, you can start and stop the services with the Windows Service dialog or command prompt, or remove the services using solidDB command line options.

Starting solidDB as a service for the first time

The first time you want to run solidDB as a service, you must first install the service, and then start the service with the Windows Service dialog or command prompt.

Before you begin

- If you have not created a database before, you must create the database by starting the server for the first time as a foreground process. This is because when solidDB is running as a service, it does not interact with a display and cannot create a new database. You can start the server as a foreground process from the command line with the command `solid` or use the **Start IBM solidDB** icon in the **Programs** menu.
- The solidDB that you intend to run as a service cannot be located on a network drive.

Procedure

1. **Allow (install) Windows to run solidDB as a service.**

   In the command prompt, issue the following command:

   ```
   solid -s"install,<name>,<fullexepath> -c<working directory>[,autostart]"
   ```

   where

   `<name>` is the service name

   `<fullexepath>` is the full path for `solid.exe`

   `<working directory>` is the full path for solidDB working directory (where your `solid.ini` and license file are located)

   `[autostart]` is an optional parameter that sets the Startup Type of the service to *Automatic*, that is, solidDB will run automatically as a service when Windows is started.

   Note:

   Regardless of the `[autostart]` parameter, the service is not started automatically at the time of install. For the first time, the service has to be started manually in the Windows Services dialog or command prompt. (See step 2 below.)

   Example 1

   The following command installs a service named SOLID (with Startup Type *Manual*) when solidDB is installed into the directory `C:\soliddb` and the working directory is `C:\soliddb`.

   ```
   solid -s"install,SOLID,C:\soliddb\bin\solid.exe -cC:\soliddb"
   ```

   Example 2

   The following command installs a service named SOLID (with Startup Type *Automatic*) when solidDB is installed into the directory `C:\soliddb` and the working directory is `C:\soliddb`. The next time Windows is started, solidDB will automatically run as a service.

   ```
   solid -s"install,SOLID,C:\soliddb\bin\solid.exe -cC:\soliddb,autostart"
   ```

   Tip:

Alternatively, you can create the service using the Windows command line utility `sc.exe`. In that case, to start solidDB in a services mode, you must include the solidDB `-sstart` command line option in the command. For example:

```
sc create SOLID binPath= "c:\soliddb\bin\solid.exe -cC:\soliddb -sstart"
```

The `-sstart` command line option is required to remove the GUI-based interactions between the solidDB server and the user. Programs running as a Windows service cannot use those.

2. **Start the service manually in the Windows Services dialog or command prompt.**
   - You can access the Windows Services dialog through Control Panel: **Control Panel** > **Administrative Tools** > **Services**.
   - In the command prompt, issue the following command:

     ```
     sc start <name>
     ```

Results

When running as an Windows service, solidDB will log warning and error messages to the Windows event log. These messages can be viewed from Windows by using the Event Viewer, available through Control Panel: **Control Panel** > **Administrative Tools** > **Event Viewer**. Messages are also logged to the `solmsg.out` file.

Starting and stopping solidDB services

The solidDB services can be started and stopped using the Windows Services dialog or command prompt.

Procedure

- You can access the Services dialog through Control Panel: **Control Panel** > **Administrative Tools** > **Services**.
- In the command prompt,
  - issue the following command to start the service:

    ```
    sc start <name>
    ```

  - issue the following command to stop the service:

    ```
    sc stop <name>
    ```

where `<name>` is the name of the service you want to start or stop.

Removing solidDB services

You can remove the solidDB services using solidDB command line options.

Procedure

1. **Stop the service in the Windows Services dialog or command prompt.**
   - You can access the Windows Services dialog through Control Panel: **Control Panel** > **Administrative Tools** > **Services**.
   - In the command prompt, issue the following command:

     ```
     sc stop <name>
     ```

   where `<name>` is the name of service you want to stop.
2. **Remove the solidDB service.**

   In the command prompt, issue the following command:

   ```
   solid -s"remove,<name>"
   ```

Example

The following command removes a service named SOLID.

```
solid -s"remove,SOLID"
```

Running several servers on one computer

In some cases, you may want to run two or more databases on one computer. For example, you may need a configuration with a production database and a test database running on the same computer.

solidDB is able to provide one database per database server, but you can start several engines each using its own database file. To make these engines use different databases, either start the engine processes from the directories your databases are located in or give the locations of configuration files by using the command line option -c *directory_name* to change the working directory. Remember to use different network listen names for each database.

Entering timed commands

solidDB has a built-in timer, which allows you to automate your administrative tasks. You can use timed commands to execute system commands, to create backups, checkpoints, and database status reports, to open and close databases, and to disconnect users and shut down servers.

To enter a timed command, edit the At parameter in the [Srv] section of the `solid.ini` file. The syntax is:

```
At = At_string
At_string ::= timed_command [, timed_command]
timed_command ::= [ day ] HH:MM command argument
day ::= sun | mon | tue | wed | thu | fri | sat
```

If the day is not given, the command is executed daily.

Example:

```
[Srv]
At = 20:30 makecp, 21:00 backup, sun 23:00 shutdown
```

Note:

The format used is HH:MM (24-hour format).

The list of valid commands is in the table below:

Table 12. Arguments and defaults for different timed commands

Command	Argument	Default
backup	backup directory	the default backup directory that is set in the configuration file
throwout	user name, all	no default, argument compulsory
makecp	no arguments	no default
shutdown	no arguments	no default

Table 12. Arguments and defaults for different timed commands (continued)

Command	Argument	Default
report	report file name	no default, argument compulsory
system	operating system command For example in Linux® environments: `cp solmsg.out solmsg2.out`	no default
open	no argument	no default
close	no argument	no default

Compacting the database files

What is database reorganization

solidDB server is capable of allocating new disk pages as the database grows. However, it does not free the space allocated previously in the database files even if it is not needed any more. Instead, it maintains a list of unused pages for later use. In some applications, however, there may be short-term peaks in the database space usage, resulting in large allocated disk space. If such peaks are seldom, there may be a need to return the unused space back to the file system. The database file reorganization feature serves this particular purpose.

How does the database reorganization work

The current implementation allows performing database file compaction in offline mode, at the page level. Offline means that a database file being compacted cannot be actively used by the server. Page level means that only empty pages are discovered and removed from the file. No intra-page compaction is performed, i.e. data is not moved among pages.

When using the feature, note that the reorganization operation may not be recoverable. If there is a failure during the reorganization run, neither the run nor the database file can be later recovered. To protect yourself against such failures, make a database backup before starting the reorganization.

Database reorganization command line options

There are two command line options available for database reorganization: *Free factor report* and *Reorganization*.

Free factor report

```
solid -x infodbfreefactor
```

The `infodbfreefactor` option outputs a report of how many free pages there are in the database, how much space is free, in kilobytes, and also a percentage value of free space. After printing the report to `ssdebug.log` and console, the solidDB process returns with a success return value.

Reorganization

```
solid -x reorganize
```

The `reorganize` option invokes database reorganization. The operation moves pages to unused slots in the database file, as long as there are any. When the page relocation is complete, the unused space is released back to the file system, i.e. the file is truncated, a new checkpoint is created, and the solidDB process terminates with a success return code. The report of the reorganization run is written to the ssdebug.log file.

See Appendix C, "solidDB command line options," on page 175 for other utilities invoked with a command line option.

Encrypting a database

By default, solidDB always encrypts passwords using the DES algorithm. If you want to encrypt also the database files and log files, you need to create an encrypted database using solidDB command line options. You can also disable the encryption of passwords.

The DES algorithm shipped with solidDB is based on a symmetric-key algorithm that uses a 56-bit key. To protect the symmetric encryption key, a startup password must be specified when creating, starting, or decrypting an encrypted database.

The solidDB DES algorithm is a weak DES algorithm that is not recommended for applications that require strong security.

Encrypting database and log files

The encryption of the entire database (database and log files) is enabled using command line options `-E` and `-x keypwdfile:<filename>`.

About this task

- The `-E` option invokes database encryption. The database can be encrypted when creating a new database or when starting an existing database.
- The `-x keypwdfile:<filename>` option provides the encryption password from a file.

  The encryption password is needed to protect the symmetric encryption key which is stored in an unencrypted header page of the database file.

  The encryption password is mandatory when `-E` is specified. The minimum length of the password is three characters. If your specify an empty password, the encryption key is left unprotected.

  Note: Alternatively, option `-S` can be used to provide the password as part of the startup command. However, this is not secure on most of systems. For example in UNIX systems, the password can be seen in the `ps` command output. Use the `-S` option only for debugging or evaluation purposes.

Procedure

- **Creating a new encrypted database**

  To create an encrypted database, include the `-E` and `-x keypwdfile:<filename>` options in the solidDB startup command.

  For example:

  ```
  solid -C mycatalog -U admin -P admin -E  -x keypwdfile:pwd.txt
  ```

- **Encrypting an existing database**

  To encrypt an existing database, include the `-E` and `-x keypwdfile:<filename>` options in the solidDB startup command.

For example:

```
solid -U admin -P admin -E  -x keypwdfile:pwd.txt
```

Starting an encrypted database

To start an encrypted database, you must provide the encryption password at the startup. If you do not include the password in the startup command, the server prompts you for the password.

Procedure

Start solidDB using the following command:

```
solid -x keypwdfile:<filename>
```

For example:

```
solid -x keypwdfile:pwd.txt
```

Alternative, you can provide the password using the `-S` command line option:

```
solid -S <password>
```

Changing the encryption password

To change the password of the encryption key, solidDB must be started using option `-E` and the options specifying the old and the new password.

Procedure

Changing the encryption password
To change the encryption password, start solidDB with the following command syntax:

```
solid -E -x keypwdfile:<old key filename> -x keypwdfile:<new key filename>
```

For example:

```
solid -E -x keypwdfile:pwd.txt -x keypwdfile:newpdw.txt
```

Alternatively, you can specify the new and old password in the command line using the `-S` option

```
solid -E -S <old_password> -S <new_password>
```

Decrypting a database

You can decrypt a database with the option `-x decrypt`. You also need to provide the encryption password.

Procedure

Decrypting a database
To decrypt a database, start solidDB with the following command syntax:

```
solid -x decrypt -x keypwdfile:<filename>
```

For example:

```
solid -x decrypt -x keypwdfile:pwd.txt
```

Querying database encryption level

You can check the database encryption level using the DATABASE_ENCRYPTION_LEVEL() function. This can be useful, for example, if your system does not allow storing data in an unencrypted file, and you need to register a new replica.

Procedure

Use the DATABASE_ENCRYPTION_LEVEL() function. The function has the following return values:

- 0 - no encryption
- 1 - encrypted, the key is not protected (empty password)
- 2 - encrypted, the key is protected by a separate startup password

Making backups of encrypted databases

Database backups and netbackups create encrypted copies of the database with the same encryption key and password.

Encrypting HotStandby servers

In High Availability (HotStandby) configurations, the Primary and Secondary servers must use the same encryption method and encryption key.

Encrypt the Primary database first and then copy or netcopy it.

HotStandby traffic is not encrypted by means of database file encryption. To protect the HSB traffic, other security means are needed. When making an HSB copy or netcopy, the database file and logs are transfered in encrypted form to avoid redundant encryption/decryption of the files.

Encryption and performance

Using an encrypted database affects the database server performance for both read and write operations.

1. On read type operations, performance impact is mostly determined by the cache hit rate and is not significant when the cache hit rate is high.
2. On insert and update operations, the server encrypts and decrypts the log files (if they are used) and in this case performance penalty can be more significant.

3 Configuring solidDB

This chapter describes how to configure solidDB to meet your environment, performance, and operation needs. It includes the most important parameters and their settings. See "Managing server-side parameters" on page 55 for step-by-step instructions on how to view and set the parameter values by using solidDB Remote Control (solcon), or SQL Editor (solsql).

Important:

If you are using solidDB with linked library access, please refer to *IBM solidDB Linked Library Access User Guide* for more information on parameters that are specific to linked library access.

If you are using solidDB with the HotStandby component, please refer to *IBM solidDB High Availability User Guide* for information on HotStandby-specific parameters.

Configuration files and parameter settings

solidDB gets most of its configuration information from the `solid.ini` file. There are two different `solid.ini` configuration files, one on the server and one on the client. Neither configuration file is obligatory. If there is no configuration file, the factory values are used. The `solid.ini` configuration files contain configuration parameters for the client and for the server, respectively. The client-side configuration file is used if the ODBC driver is used and the file must be located in the working directory of the application.

Note:

In solidDB documentation, references to `solid.ini` file are usually for the server-side `solid.ini` file.

When solidDB starts, it attempts to open `solid.ini` first from the directory set by the SOLIDDIR environment variable. If the file is not found from the path specified by this variable or if the variable is not set, the server or client attempts to open the file from the current working directory. The current working directory is normally the same as the directory from which you started the solidDB server, or a client application. You may specify a different working directory by using the -c server command-line option. For more information about command line options, see the section Appendix C, "solidDB command line options," on page 175.

The configuration files contain settings for the solidDB parameters. If a value for a specific parameter is not set in the `solid.ini` file, solidDB will use a factory value for the parameter. The factory values may depend on the operating system you are using.

Generally, factory values offer good performance and operability, but in some cases modifying some parameter values can improve performance.

You can modify the configuration by setting parameter name/value pairs in the `solid.ini` file. For example, to specify the network address of the server, you use the parameter name Listen and an appropriate value, for example,

```
Listen=tcp 192.168.255.1 1315
```

This specifies that when the server listens for client requests, it should listen using the TCP/IP protocol, the network address 192.168.255.1, and the port number 1315.

Parameters are grouped according to section categories in the configuration file. For an overview of the section categories and all the available parameters, see the sections Appendix A, “Server-side configuration parameters,” on page 117 and Appendix B, “Client-side configuration parameters,” on page 171.

Each section category starts with a section name inside square braces, for example:

```
[com]
```

The [com] section lists communication information. Note that section names are case insensitive. The section names "[COM]", "[Com]", and "[com]" are equivalent.

Below is a sample section from a server-side `solid.ini` configuration file:

```
[IndexFile]
FileSpec_1=C:\soldb\solid1.db 1000M
CacheSize=64M
```

Most important client-side parameters

This section describes the most important solidDB client-side parameters and their default settings.

Defining network names (Com section)

A client application uses a network name to specify which protocol to use when communicating with the server, and which server to connect to.

Connect parameter

The Connect parameter in the [Com] section defines the default network name (connect string) for a client to connect to when it communicates with a server. Not surprisingly, since the client should talk to the same network name as the server is listening to, the value of the Connect parameter on the client should match the value of the Listen parameter on the server.

The default value is Operating System dependent. Refer to 6, “Managing network connections,” on page 99.

The following connect line tells the client to communicate with the server by using the TCP/IP protocol to talk to a computer named 'spiff' using server port number '1313'.

```
[Com]
connect = tcpip spiff 1313
```

When an application program is using a solidDB ODBC Driver, the ODBC Data Source Name is used and the Connect parameter has no effect.

Note that similar connect parameters are used in sections [HotStandby] and [Synchronizer] to enable connections between solidDB servers. For the description of these parameters, refer to *IBM solidDB High Availability User Guide* and *IBM solidDB Advanced Replication User Guide*.

Format of the connect string

The same format of the connect string applies to all listen configuration parameters as well as to connect strings used in ODBC applications.

Connect string format:

```
protocol_name [options] [server_name] [port_number]
```

where options can be any number of:

Table 13. Connect string options

Option	Meaning
-z	Data compression is enabled for this connection
-c *milliseconds*	Login timeout is specified (the default is operating-system-specific). A login request fails after the specified time has elapsed. Note: applies for the tcp protocol only.
-r *milliseconds*	Connection (or read) timeout is specified (the default is 60 s). A network request fails when no response is received during the time specified. The value 0 sets the timeout to infinite. Note: applies for the tcp protocol only.

Examples:

```
tcp localhost 1315
tcp 1315
tcp -z -c1000 1315
nmpipe host22 SOLID
```

Trace parameter

If you change the **Trace** parameter default setting from No to Yes, solidDB starts logging trace information on network messages for the established network connection to the default trace file or to the file specified in the **TraceFile** parameter.

TraceFile parameter

If the **Trace** parameter is set to Yes, then trace information on network messages is written to a file specified by the **TraceFile** parameter. If no file name is specified, the server uses the default value `soltrace.out`, which is written to the current working directory of the server or client, depending on which end the tracing is started at.

Most important server-side parameters

This section describes the most important solidDB server-side parameters and their default settings.

Defining network names (Com section)

When a server is started, it will start listening to one or more protocols with network names that distinguish it in the network. A client application uses a similar network name to specify which protocol to use and which server to connect to.

Listen parameter

The **Listen** parameter in the [Com] section defines the network name for the server; this is the protocol and name that a solidDB server uses when it starts to listen to the network. Client processes communicate with the server using this network name. The default value is operating system dependent. Refer to 6, “Managing network connections,” on page 99, for details on the parameter format.

```
[Com]
Listen = tcpip localhost 1313
```

Managing database files and caching (IndexFile section)

In solidDB, data and indexes are stored in the same file(s). The term "index file" is used as a synonym for the term "database file". The [IndexFile] section of the `solid.ini` file contains parameters that specify the name and location of the file(s) used to store the database. The [IndexFile] section of `solid.ini` also controls the caching-related parameters.

FileSpec_[1...n] parameter

The **FileSpec** parameter describes the location and the maximum size of an index file (database file). To define the location and maximum size, the **FileSpec** parameter accepts the following three arguments:

- database file name
- max filesize
- device number (optional)

```
[IndexFile]
 FileSpec_1=SOLID.DB 2000M
```

The default value for this parameter is

solid.db 2147483647

(which equals 2 GB-1 expressed in bytes)

The size unit is 1 byte. You can use *K* and *M* unit symbols to denote kilobytes and megabytes, respectively. The maximum file size is (4G-1)*blocksize. With the default 8 KB block size, this makes 32 TB - 1.

The FileSpec parameter is also used to divide the database into multiple files and onto multiple disks. To divide the database into multiple files, specify another **FileSpec** parameter identified by the number 2. The index file will be written to the second file if it grows over the maximum value of the first **FileSpec** parameter.

In the following example, the parameters divide the database file on the disks C:, D: and E: to be split after growing larger than about 1 GB (=1073741824 bytes). This example does not use the optional device number.

```
[IndexFile]
FileSpec_1=C:\soldb\solid.1 1000M
FileSpec_2=D:\soldb\solid.2 1000M
FileSpec_3=E:\soldb\solid.3 1000M
```

Note:

The index file locations entered must be valid path names in the server's operating system. For example, if the server runs on a UNIX operating system, path separators must be slashes instead of backslashes.

Although the database files reside in different directories, the file names must be unique. In the above example, the different device numbers indicate that C:, D: and E: partitions reside on separate disks.

There is no practical limit to the number of database files you may use.

Splitting the database file on multiple disks will increase the performance of the server because multiple disk heads will provide parallel access to the data in your database.

Note that you may need to have multiple files on a single disk if your physical disk is partitioned into multiple logical disks and no single logical disk can accommodate the size of the database file you expect to create.

If the database file is split into multiple physical disks, then multithreaded solidDB is capable of assigning a separate disk I/O thread for each device. This way the server can perform database file I/O in a parallel manner. Read section *Dedicated threads* in "Types of threads" on page 7 for more details.

The optional "device number" that you may specify for each data file helps the server optimize its performance. Note that the actual device number serves only as a means for you to designate a distinct number for each physical device; the device number serves no other purpose, such as indicating the brand, model, or characteristics of your storage device.

If you have different files on the same physical device, use the same device number for each of those files. For example, assume that your computer runs Microsoft Windows and has two physical disk drives. The first physical disk drive is *C:*. The second physical disk drive is partitioned into two logical disk drives, *D:* and *E:*. If one data file is put on *C:*, one on *D:*, and one on *E:*, then the `solid.ini` file might look like the following:

```
FileSpec_1=C:\soldb\solid.1 1000M 1
FileSpec_2=D:\soldb\solid.2 1000M 2
FileSpec_3=E:\soldb\solid.3 1000M 2
```

In this case, *FileSpec_2* and *FileSpec_3* use the same physical device (even though the device names D: and E: are different), so they are assigned the same device number. The actual values used for the device number (1 for *C:*, 2 for *D:*, and 2 for *E:*) are arbitrary and meaningless.

If your database has reached the maximum size specified by the **FileSpec** parameter, you can increase the limit. Simply shut down the server, increase the size field, and restart the server. You may increase the size this way, but you must not try to decrease the size this way.

CAUTION:
Do not attempt to use the FileSpec parameter to decrease the size of a database; you risk losing pre-existing data and corrupting the database.

CacheSize

The **CacheSize** parameter defines the amount of main memory used to maintain the shared buffer pool of the disk database. This buffer pool is called the database cache. The factory value depends on the server operating system. For the pure in-memory database operation, the cache size is mostly irrelevant once it is not less than 8 MB. The absolute minimum size is 512 kilobytes. For example:

```
[IndexFile]
CacheSize=512
```

The size unit is bytes. You may also specify the amount of space in units of megabytes, for example, "10M" for 10 megabytes. Although solidDB is able to run with a small cache size, a larger cache size generally speeds up the server. The cache size needed depends on the size of the database, the number of connected users, and the nature of the operations executed against the server.

The default cache size is 32 MB.

Specifying the local backup directory (General section)

Backups of the database, log files and the configuration file `solid.ini` are copied to the local backup directory. The directory must exist and it must have enough disk space for the backup files since all the database files of one database are copied to the same directory. It can be set to any existing directory except the solidDB database file directory, the log file directory or the working directory.

BackupDirectory parameter

The **BackupDirectory** parameter in the [General] section defines a name and location for your backup directory. Note that default 'backup' is a directory relative to your solidDB working directory. For example, if the parameter is:

```
[General]
BackupDirectory=backup
```

then the backup will be written to a directory that is a sub-directory of the solidDB directory.

Note:

The backup directory entered must be a valid path name in the server's operating system. For example, if the server runs on a UNIX operating system, path separators must be slashes instead of backslashes.

Specifying the network backup directory (General section)

The target directory in the NetBackup Server for the backup files, log files and the configuration file is set with the **NetBackupDirectory** parameters in the source server and the network server side. If the remote directory doesn't exist, it is created if possible.

Source-side parameter

The parameter

```
[General]
NetBackupDirectory=netbackupdir
```

in the source server sets the remote directory for use of Network Backup. The *netbackupdir* is either absolute or relative to the root directory of the NetBackup Server.

Netbackup server-side parameter

The parameter

```
[Srv]
NetBackupRootDir=netbackup root dir
```

in the NetBackup Server sets the root directory to all netbackup operations using relative path expressions by their NetBackupDirectory specifications. The *netbackup root dir* is either absolute or relative to the working directory.

Important:

NetBackup copies logical database consisting of multiple files to one flat file to the NetBackupDirectory by default. Instead of flattening the structure to one file you can define multiple files to which the source database files are mapped in netbackup. Mapping source database file(s) to multiple backup database files is done by way of using the `backup.ini` file.)

To ensure the durability of committed transactions, transaction results are written immediately to a file in a specified directory when the transaction is committed. This file must be stored to a local drive using local disk names to avoid problems with network I/O and to achieve better performance. The default log file directory is the solidDB working directory.

FileNameTemplate

The FileNameTemplate parameter in the Logging section defines a filename structure for the transaction log files. For example, the following setting

```
[Logging]
FileNameTemplate = d:\logdir\sol#####.log
```

instructs solidDB to create log files to directory `d:\logdir` and to name them sequentially starting from `sol00001.log`.

Note:

Placing log files on a physical disk separate from database files improves performance.

The filename can also be structured by using the FileNameTemplate parameter together with the LogDir parameter, in which case the LogDir parameter defines the directory prefix of the filename and the FileNameTemplate parameter defines the actual filename. For more information, see "Logging section" on page 142.

Specifying a directory for the external sorter algorithm (Sorter section)

The external sorter algorithm is used for sorting tasks that do not fit in main memory. When the TmpDir_[1...N] is specified in the configuration file, the external sorter algorithm is enabled. All temporary files used by the external sort are created in a specified directory (or directories) and are automatically deleted.

Note that an "external sort" requires space both on disk and in memory, not just space on the disk. You can configure the maximum amount of disk space to use by setting the **MaxMemPerSort** and **MaxCacheUsePercent** parameters in the [Sorter] section of the `solid.ini` file.

TmpDir_[1...N]

The TmpDir[1-N] parameter in the Sorter section defines the directory (or directories) that can be used by the external sorter. There is no default setting. For example:

```
[Sorter]
TmpDir_1=c:\soldb\temp.1
TmpDir_2=d:\soldb\temp.2
TmpDir_3=g:\soldb\temp.3
```

To achieve better performance, these files must be stored to a local drive using local disk names to avoid network I/O. Note that when temporary directories are not defined, this can lead to poor query performance.

Setting threads for processing (Srv section)

In addition to the communication, I/O, and log manager threads, solidDB can start general purpose worker threads to execute user tasks in the server's tasking system. Read “Multithread processing” on page 7 for more details.

The optimum number of threads depends on the number of processors the system has installed. Usually it is most efficient to have between two and eight threads per processor.

You must experiment to find the value that provides the best performance on your hardware and operating system. A good formula to start with is:

threads= (2 x number of processors) + 1

Threads

The Threads parameter in the [Srv] section defines the number of general purpose worker threads used by solidDB. For example:

```
[Srv]
Threads=9
```

Setting SQL trace level (SQL section)

The SQL Info facility lets you specify a tracing level on the SQL Parser and Optimizer. For details on each level, see *IBM solidDB SQL Guide*.

Info

The SQL Info facility is turned on by setting the Info parameter to a non-zero value in the [SQL] section of the configuration file. The output is written to a file named `soltrace.out` in the solidDB directory.

Use this parameter for troubleshooting purposes only as it slows down the server performance significantly. This parameter is typically used for analyzing performance for a specific single query or specific queries. Standard solidDB monitoring is a better choice for generic application SQL database tracing.

Specifying network communication tracing (Com section)

The communication tracing facility is necessary, for instance, if the network hardware is not functioning properly. By turning the tracing on, the communication layer is capable of logging even the system specific errors and may help in diagnosing the real problem in the network. For details, read “The network trace facility” on page 111. The following parameters control the outputting of network trace information.

Trace

If you change the **Trace** parameter default setting from No to Yes, solidDB starts logging trace information on network messages for all the established network connections to the default trace file or to the file specified in the **TraceFile** parameter.

TraceFile

If the **Trace** parameter is set to Yes, then trace information on network messages is written to a file specified by the **TraceFile** parameter. If no file name is specified, the server uses the default value `soltrace.out`, which is written to the current working directory of the server or client, depending on which end the tracing is started at.

Managing server-side parameters

You can view and modify solidDB parameters and their values in the following ways:

- Entering the commands:
  ADMIN COMMAND 'parameter'
  and
  ADMIN COMMAND 'describe parameter'
  in solidDB SQL Editor (teletype).
- Directly, by editing the `solid.ini` file in the solidDB directory.

The sections below contain instructions for managing parameters with ADMIN COMMAND and `solid.ini`.

Note: For details on viewing and setting server communication protocol parameters only, see the section 6, “Managing network connections,” on page 99.

Viewing and setting parameters with ADMIN COMMAND

With ADMIN COMMAND, you can change the parameters remotely through a solidDB server without restarting it. All parameters are accessible even if they are not present in the `solid.ini` configuration file. If the parameter is not present, the factory value is used.

Viewing parameters

A summary view of many parameters or one parameter may be obtained with the command

```
ADMIN COMMAND 'parameter [-r] [section_name[.parameter_name]]';
```

where:

- -r option specifies that only the current value is required
- *section_name* is the category name where the parameter is located in `solid.ini`

To view all parameters, enter the following command in solidDB SQL Editor (teletype):

ADMIN COMMAND 'parameter';

A list of all parameters with *current*, *startup value*, and *factory values* is returned. You can restrict the viewed parameters to a specific section by adding a section name, e.g.:

```
ADMIN COMMAND 'parameter logging';
```

You can view the values related a single parameter by giving a full parameter name, like in:

```
admin command 'parameter logging.durabilitylevel';
    RC TEXT
    -- ----
    0 Logging DurabilityLevel 3 2 2
1 rows fetched.
```

The three values shown are (in this order):

- *current value*
- *startup value* that was used when the server was started up
- *factory value* preset in the product

If desired, you can also qualify this command with a -r option to display only the current values. For example:

ADMIN COMMAND 'parameter -r';

Viewing the description of a specific parameter

You can also view a more detailed description of a specific parameter, which includes valid parameter types and access modes. This is useful information, especially because parameters may need to be handled dynamically; parameter support may vary between products, platforms, or releases.

To view a parameter's description, enter the following command using solidDB SQL Editor (teletype):

```
ADMIN COMMAND 'describe parameter [section_name[.parameter_name]] ';
```

A result set for a single parameter looks like this:

```
admin command 'describe parameter logging.durabilitylevel';
    RC TEXT
    -- ----
    0 DurabilityLevel
    0 Default transaction durability level
    0 LONG
    0 RW
    0 2
    0 3
    0 2
7 rows fetched.
```

The rows of the resultset are:

- *Parameter name* is the name of the parameter, for example **CacheSize**.
- *Description* of the parameter
- *Data type*
- *Access mode* that may be one of the following:
  - RO: read-only, the value cannot be changed dynamically
  - RW: read/write, the value may be changed dynamically and the change takes effect immediately
  - RW/STARTUP: the value may be changed dynamically but the change takes effect upon next server startup
  - RW/CREATE: the value may be changed dynamically but the change takes effect when a new database is created

- *Startup value* displays the parameter's startup value
- *Current value* displays the parameter's current value
- *Factory value* displays the value preset in the product

Setting a parameter value

To set a value for a specific parameter, enter the following command using solidDB SQL Editor (teletype):

```
ADMIN COMMAND 'parameter section_name.parameter_name=value [temporary]';
```

where:

value is a valid parameter value.

Note:

If no value is specified, this sets the parameter with a factory (or unset) value. Furthermore, if you assign a parameter value with an asterisk (*), the parameter will be set to its factory value.

When temporary is set, the changed value is not stored in the `solid.ini` file.

Note that, optionally, you can provide blanks around the equal sign.

Example:

```
--set communication trace on
ADMIN COMMAND 'parameter com.trace = yes';
```

Note:

Parameter management operations are not part of a transaction and cannot be rolled back.

The commands return the new value as the resultset. If the parameter's access mode is RO (read-only) or the value entered is invalid, the ADMIN COMMAND statement returns an error.

Persistence of parameter modifications

All the changes made to parameters having the access mode RW* are stored in the `solid.ini` file at the next checkpoint. This does not apply to values set with the temporary option.

It is also possible to request an immediate storing of changed values, with the command:

```
ADMIN COMMAND 'save parameters [ini_file_name]';
```

When *ini_file_name* is not specified, the current `solid.ini` file is re-written. Otherwise, a full configuration file is written to a new location. This is a convenient way to save configuration file checkpoints for later use.

Viewing and setting parameters in `solid.ini`

1. Open the `solid.ini` file located in the working directory of your solidDB process.
2. View the value of the parameter.

The parameters displayed are the parameters currently active in the server. If you have not set a parameter value, the factory value is used at startup. The factory value may depend on the operating system that solidDB runs on.

3. If necessary, add the section, the parameter, and the parameter's value.
4. Save the changes.

You must restart the server to activate the changes.

Constant parameter values

The parameter access mode for the Blocksize parameter in the IndexFile section of the configuration file is RO. The parameter is set when the database is created and cannot be modified afterwards.

If you want to use a different constant value, you have to create a new database. Before creating a new database, set the new parameter constant value by editing the `solid.ini` file in the solidDB directory.

The following example sets a new block size for the index file by adding the following lines to the `solid.ini` file :

```
[IndexFile]
Blocksize = 4096
```

After editing and saving the `solid.ini` file, move or delete the old database and log files, and start solidDB.

Note:

The log block size can be changed between startups of the server.

4 Using solidDB data management tools

solidDB data management tools are a set of utilities for performing various database tasks. The tools are:

- solidDB Remote Control (solcon) and solidDB SQL Editor (solsql) for command line sessions at the operating system prompt.
- solidDB Speed Loader (solload) for loading data from external ASCII files into a solidDB database.
- solidDB Export (solexp) for unloading data from a solidDB database to ASCII files.
- solidDB Data Dictionary (soldd) for retrieving data definition statements from a solidDB database.

Note: solidDB data management tools do not support the Transparent Failover (TF) feature. Transparent Failover is a characteristic of the High Availability configuration. It hides the server change from the user. For more information, refer to *IBM solidDB High Availability User Guide*.

Entering password from a file

User-identification information is typically entered as plain text, for example to solidDB startup command, and to solidDB data management tools. It is, however, possible to enter password from a file. This way the password can't be seen by running the UNIX command ps.

The syntax is as follows:

```
command -x pwdfile:filename
```

The command can be any of the following: solcon, soldd, solexp, solid, solload, solsql. Option *filename* can be either absolute or relative to the working directory.

The first character string ending at newline character is read and considered as password. Preceding space and newline characters are ignored. If the password includes space or newline characters, it must be enclosed in quotes. Using quotes, however, means that quote and backslash characters that belong to the password must be escaped by a backslash character.

Command examples:

```
solsql -x pwdfile:userpwd "tcp solsrv 1313" dba
solid -f -c soldb -x pwdfile:solpwd -U dba
```

solidDB Remote Control (solcon)

With solidDB Remote Control, you can execute administrative commands (equivalent to the solidDB SQL ADMIN COMMANDs), at the command line, command prompt, or by executing a script file that contains the commands.

Note: The user performing the administration operation must have SYS_ADMIN_ROLE or SYS_CONSOLE_ROLE rights, or the connection will be refused.

Starting solidDB Remote Control

Start solidDB Remote Control by issuing the command solcon at the operating system prompt.

You can also specify the following syntax and include these optional command line arguments:

```
solcon options servername username password
```

where *options* can be:

Table 14. solcon command options

Option Syntax	Description
-c *dir*	Change working directory.
-e *command string*	Execute the specified Remote Control command.
-f *filename*	Execute command string from a script file.
-x pwdfile: *filename*	Read password from the filename.
-h, -?	Help = Usage.

Servername is the network name of a solidDB server that you are connected to. Logical Data Source Names can also be used with tools; refer to 6, “Managing network connections,” on page 99 for further information. The given network name must be enclosed in quotes.

Username is required to identify the user and to determine the user's authorization. Without appropriate rights, command execution is denied.

Password is the user's password for accessing the database.

solidDB Remote Control connects to the first server specified in the Connect parameter in the `solid.ini` file. If you specify no arguments, you are prompted for the database administrator's user name and password. You can give connection information at the command line to override the connect definition in `solid.ini`.

To exit Remote Control, enter the command exit.

Remote control

Start up Remote Control with the server name and the administrator's username and password:

```
solcon "tcp localhost 1313" admin iohi4y
```

Start up Remote Control to back up a specific database:

```
solcon -ebackup 'tcpip 1313" dbadmin password
```

Entering commands in solidDB Remote Control

After the connection to the server is established, the command prompt appears.

You can execute all commands at the command line with the -e option or in a text file with the -f option. You can also execute administrative commands programmatically using options of the SQL command "ADMIN COMMAND".

When you execute administrative commands in solidDB Remote Control, you provide only the command_name as the syntax for the command string (without quotes); for example, the SQL command ADMIN COMMAND 'backup' in solidDB Remote Control is simply:

```
backup
```

For a list of administrative commands you can use in solidDB Remote Control, refer to the description of "ADMIN COMMAND" in the "solidDB SQL Syntax" appendix in *IBM solidDB SQL Guide*.

When there is an error in the command line, solidDB Remote Control gives you a list of the possible options as a result. Please be sure to check the command line you entered.

Table 15. Remote control specific commands

Command	Abbreviation	Explanation
exit	ex	Exits solidDB Remote Control.
help	?	Displays available Remote Control commands.

solidDB SQL Editor (solsql)

With solidDB SQL Editor, SQL statements (including the SQL ADMIN COMMANDs) can be issued at the command line, command prompt, or by executing a script file that contains the SQL statements. For a formal definition of SQL statements and a list of ADMIN COMMANDs, refer to the description of "ADMIN COMMAND" in the "Solid® SQL Syntax" appendix in *IBM solidDB SQL Guide*. To access a short description of available ADMIN COMMANDs, including short abbreviations, execute:

ADMIN COMMAND 'help'

Starting solidDB SQL Editor

Start solidDB SQL Editor by issuing the command solsql at the operating system prompt.

You can also specify the following syntax and include these optional command line arguments:

```
solsql options servername username password
```

where options can be:

Table 16. solsql command options

Option Syntax	Description
-a	Auto commit every statement.

Table 16. solsql command options (continued)

Option Syntax	Description
-c *dir*	Change working directory.
-e *sql-string*	Execute the SQL string; if used commit can only be done using -a.
-f *filename*	Execute SQL string from a script file.
-h, -?	Help = Usage.
-m	Expect input in multi-byte character format.
-o *filename*	Write result set to this file.
-O *filename*	Append result set to this file.
-s *schema_name*	Use only this schema.
-t	Print execution time per command.
-u	Expect input in UTF-8 format.
-x pwdfile: *filename*	Read password from the filename.
-x onlyresults	Print only rows.
-x outputsql	This command-line switch also prints out the executed SQL commands instead of only printing out the results of each operation.
-x returnerroronexit	This command-line switch is used to display return codes for SQL errors and user raised procedure errors. The possible return codes are: Code 60 is returned if the execution of an SQL statement fails. Code 61 is returned if a procedure call returns an error. If several sql statements and/or procedure calls fail during the execution of an SQL script, the returned code is that of the first failure.
-x stoponerror	This command-line switch is used to force stop and exit solsql immediately when an error is detected.

Note:

If the user name and password are specified at the command line, the server name must also be specified. Also if the name of the SQL script file is specified at the command line (except with the -f option), the server name, user name, and password must also be specified. Remember to commit work at the end of the SQL script or before exiting SQL Editor.

Servername is the network name of a solidDB server that you are connected to. Logical Data Source Names can also be used with tools; Refer to 6, “Managing network connections,” on page 99 for further information. The given network name must be enclosed in double quotes.

Username is required to identify the user and to determine the user's authorization. Without appropriate rights, command execution is denied.

Password is the user's password for accessing the database.

solidDB SQL Editor connects to the first server specified in the Connect parameter in the `solid.ini` file. If you specify no arguments, you are prompted for the database administrator's user name and password.

When there is an error in the command line, the solidDB SQL Editor gives you a list of the possible options as a result. Please be sure to check the command line you entered.

To exit SQL Editor, enter the command exit.

Running SQL scripts

You can execute SQL scripts directly in the solidDB SQL Editor. The SQL script that you specify can also call other SQL scripts. The syntax for script calls in SQL Editor is:

```
@filename
```

For example:

```
---Execute the SQL script named "insert_rows.sql" in the
-- root ("\") directory of the C: drive.
@\c:\insert_rows.sql;
```

Both absolute and relative path names are supported. If you specify a relative path, it should be relative to the SQL Editor working directory.

SQL script examples

Assuming that a database connection is established, this command example executes the SQL statements terminated by a semicolon:

```
create table testtable (value integer, name varchar);
commit work;
```

Start SQL Editor and execute the tables.sql script:

```
solsql "tcp localhost 1313" admin iohe47 tables.sql
```

Executing SQL statements with solidDB SQL Editor

After the connection to the server has been established, a command prompt appears. solidDB SQL Editor executes SQL statements terminated by a semicolon.

Example:

```
create table testtable (value integer, name varchar);
commit work;

insert into testtable (value, name) values (31, 'Duffy Duck');
select value, name from testtable;
commit work;

drop table testtable;
commit work;
```

Executing a SQL script from a file

To execute a SQL script from a file, the name of the script file must be given as a command line parameter:

```
solsql servername username password filename
```

All statements in the script must be terminated by a semicolon. solidDB SQL Editor exits after all statements in the script file have been executed.

Example:

```
solsql "tcp localhost 1313" admin iohe4y tables.sql
```

Note:

Remember to commit work at the end of the SQL script or before exiting solidDB SQL Editor. If an SQL string is executed with the option -e, commit can only be done using the -a option.

solidDB Speed Loader (solload)

solidDB Speed Loader (solload) is a tool for loading data from external ASCII files into a solidDB database. solidDB Speed Loader can load data in a variety of formats and produce detailed information of the loading process into a log file. The format of the import file, that is, the file containing the external ASCII data, is specified in a control file.

The data is loaded into the database through the solidDB program. This enables online operation of the database during the loading. The data to be loaded does not have to reside in the server computer.

Please note the following:

- The table must exist in the database in order to perform data loading.
- Catalog support is available in solidDB Speed Loader. The following syntax is supported:

  catalog_name.schema_name.table_name
- solidDB Speed Loader checks for the following constraints:
  - referential
  - NOT NULL
  - unique
- solidDB Speed Loader does not support check constraints, which are used to specify data value restrictions in columns and are defined using the CREATE TABLE and ALTER TABLE statement.

However, solidDB Speed Loader always checks for unique or foreign key constraints that are defined using the CREATE TABLE statement. For more details on constraints, see the CREATE TABLE syntax in the *Appendix: solidDB SQL Syntax* in the *IBM solidDB SQL Guide*.

Control file

The control file provides information on the structure of the import file. It gives the following information:

- name of the import file
- format of the import file
- table and columns to be loaded

Note: Each import file requires a separate control file. solidDB Speed Loader loads data into one table at a time.

For more details about the control file format, read “Control file syntax” on page 67.

Import file

The import file must be of ASCII type. The import file may contain the data either in a fixed or a delimited format:

- In fixed-length format data records have a fixed length, and the data fields inside the records have a fixed position and length.
- In delimited format data records can be of variable length. Each data field and data record is separated from the next with a delimiting character such as a comma (this is what solidDB Export produces). Fields containing no data are automatically set to NULL.

Data fields within a record may be in any order specified by the control file. Please note the following:

- Data in the import file must be of a suitable type. For example, numbers that are presented in a float format cannot be loaded into a field of integer or smallint type.
- Data of varbinary and long varbinary type must be hexadecimal encoded in the import file.
- When using any fixed-width field, regardless of the data type, Solload expects the import file to have the specified width, even when NULL is used.

Message log file

During loading, solidDB Speed Loader produces a log file containing the following information:

- Date and time of the loading
- Loading statistics such as the number of rows successfully loaded, the number of failed rows, and the load time if it has been specified with the option
- Any possible error messages. For details on solidDB Speed Loader errors, see “solidDB Speed Loader (solload) errors” on page 255.

If the log file cannot be created, the loading process is terminated. By default the name of the log file is generated from the name of the import file by substituting the file extension of the import file with the file extension .log. For example, `my_table.ctr` creates the log file `my_table.log`. To specify another file name, use the option -l.

Configuration file

A configuration file is not required for solidDB Speed Loader. The configuration values for the server parameters are included in the solidDB configuration file `solid.ini`.

Client copies of this file can be made to provide connection information required for solidDB Speed Loader. If no server name is specified in the command line, solidDB Speed Loader will choose the server name it will connect to from the server configuration file. For example to connect to a server using the NetBIOS protocol and with the server name solidDB, the following lines should be included in the configuration file:

```
[Com]
Connect=netbios SOLIDDB
```

Starting solidDB Speed Loader

Start solidDB Speed Loader with the command solload followed by various argument options. If you start solidDB Speed Loader with no arguments, you will see a summary of the arguments with a brief description of their usage. The command line syntax is:

```
solload [options] [servername] username [password]control_file
```

where options can be:

Table 17. solload command options

Option Syntax	Description
-b *records*	Number of records to commit in one batch
-c *dir*	Change working directory
-C *catalog_name*	Set the default catalog from where data is read from or written to.
-l *filename*	Write log entries to this file.
-L *filename*	Append log entries to this file.
-n *records*	Insert array size (network version).
-s *schema_name*	Set the default schema.
-t	Print load time.
-h	Help = Usage.
-x emptytable	Load data only if there are no rows in the table.
-x errors: *count*	Maximum error count.
-x nointegrity	No integrity checks during load.
-x pwdfile: *filename*	Read password from the file.
-x skip: *records*	Number of records to skip.
-x *utf8*	WCHAR data is in UTF-8 format.

For details on the *control_file*, read the following section.

Servername is the network name of a solidDB server that you are connected to. Logical Data Source Names can also be used with tools; Refer to 6, “Managing network connections,” on page 99 for further information. The given network name must be enclosed in quotes.

Username is required to identify the user and to determine the user's authorization. Without appropriate rights, execution is denied.

Password is the user's password for accessing the database.

When there is an error in the command line, the solidDB Speed Loader gives you a list of the possible options as a result. Please be sure to check the command line you entered.

Control file syntax

The control file syntax has the following characteristics:

- keywords must be given in capital letters
- comments can be included using the standard SQL double-dash (--) comment notation
- statements can continue from line to line with new lines beginning with any word

solidDB Speed Loader reserved words must be enclosed in quotes if they are used as data dictionary objects, that is, table or column names. The following list contains all reserved words for the solidDB Speed Loader control file:

Table 18. Speed Loader reserved words

Speed Loader Reserved Words			
AND	ANSI	APPEND	BINARY
BLANKS	BY	CHAR	CHARACTERSET
DATA	DATE	DECIMAL	DOUBLE
ENCLOSED	ERRORS	FIELDS	FLOAT
IBMPC	INFILE	INSERT	INTEGER
INTO	LOAD	LONG	MSWINDOWS
NOCNV	NOCONVERT	NULLIF	NULLSTR
NUMERIC	OPTIONALLY	OPTIONS	PCOEM
POSITION	PRECISION	PRESERVE	REAL
REPLACE	SCAND7BIT	SKIP	SMALLINT
TABLE	TERMINATED	TIME	TIMESTAMP
TINYINT	VARBIN	VARCHAR	WHITESPACE

The control file begins with the statement LOAD [DATA] followed by several statements that describe the data to be loaded. Only comments or the OPTIONS statement may optionally precede the LOAD [DATA] statement.

Table 19. Full syntax of the control file

Syntax Element	**Definition**
control_file	*::= [option_part]* *load_data_part* *into_table_part* *fields* *column_list*

Table 19. Full syntax of the control file (continued)

Syntax Element	Definition
option_part	::= OPTIONS (*options*)
options	*::= option [, option]*
option	::= [SKIP = *int_literal*] [ERRORS = *int_literal*]
load_data_part	::= LOAD [DATA] [*characterset_specification*] [DATE *date_mask*] [TIME *time_mask*] [TIMESTAMP *timestamp_mask*] [INFILE *filename*] [PRESERVE BLANKS]
characterset_specification	::= CHARACTERSET { NOCONVERT | NOCNV | ANSI | MSWINDOWS | PCOEM | IBMPC | SCAND7BIT } Note that UTF8 is not allowed inside the control file.
into_table_part	::= INTO TABLE *tablename*
fields	::= [FIELDS {*termination* | *enclosure*}]
termination	::= TERMINATED BY *termination_char* [[OPTIONALLY] *enclosure*]
termination_char	::= WHITESPACE | 'char' | "char" | *hex_literal*
enclosure	::= ENCLOSED BY *enclose_char* [AND *enclose_char*]
enclose_char	::='char' | "char" | *hex_literal*
hex_literal	::= X'hex_byte_string'
column_list	*::= column [, column]*
column	*::= column_name datatype_spec* [POSITION (*int_literal* {: | -} *int_literal*)] [DATE *date_mask*] [TIME *time_mask*] [TIMESTAMP *timestamp_mask*] [NULLIF BLANKS | NULLIF NULLSTR| NULLIF '*string*' | NULLIF ((*int_literal* {: | -} *int_literal*) = '*string*')]
datatype_spec	::= {BINARY | CHAR [(*length*)] | DATE | DECIMAL [(*precision* [, *scale*])] | DOUBLE PRECISION | FLOAT [(*precision*)] | INTEGER | LONG VARBINARY | LONG VARCHAR | NUMERIC [(*precision* [, *scale*])] | REAL | SMALLINT | TIME | TIMESTAMP [(*timestamp precisionv*)] | TINYINT | VARBINARY | VARCHAR [(*length*)] }

CHARACTERSET

The CHARACTERSET keyword is used to define the character set used in the input file. If the CHARACTERSET keyword is not used or if it is used with the parameter NOCONVERT or NOCNV, no conversions are made. Use the parameter ANSI for the ANSI character set, MSWINDOWS for the Microsoft Windows character set, PCOEM for the ordinary PC character set, IBMPC for the IBM® PC character set, and SCAND7BIT for the 7-bit character set containing Scandinavian characters.

Note: UTF-8 is not allowed inside the control file.

DATE, TIME, and TIMESTAMP

The DATE, TIME and TIMESTAMP keywords can be used in two places with different functionality:

- When one of these keywords is used as a part of the load-data-part element, it defines the format used in the import file for inserting data into any column of that type.
- When a keyword appears as a part of a column definition it specifies the format used when inserting data into that column.

Note:

1. Masks used as part of the load-data-part element must be in the following order: DATE, TIME, and TIMESTAMP. Each is optional.
2. Data must be of the same type in the import-file, the mask, and the column in the table into which the data is loaded.

Table 20. Data masks

Data Type	Available Data Masks
DATE	YYYY/YY-MM/M-DD/D
TIME	HH/H:NN/N:SS/S
TIMESTAMP	YYYY/YY-MM/M-DD/D HH/H:NN/N:SS/S

In the above table, year masks are YYYY and YY, month masks MM and M, day masks DD and D, hour masks HH and H, minute masks NN and N, and second masks SS and S. Masks within a date mask may be in any order; for example, a date mask could be 'MM-DD-YYYY'. If the date data of the import file is formatted as 1995-01-31 13:45:00, use the mask YYYY-MM-DD HH:NN:SS.

Date Example in Control File

Note that the following example uses the POSITION keyword. For details on this keyword, read “POSITION” on page 74.

```
OPTIONS(SKIP=1)

LOAD DATA
RECLEN 12
INTO TABLE SLTEST2
(
    ID    POSITION(1:2) NULLIF BLANKS,
    DT    POSITION(3:12) DATE 'DD.MM.YYYY' NULLIF ((4:6) = '   ')
)
```

Date, Time, and Timestamp Examples in Control File

Note that the following example uses the FIELDS TERMINATED BY keyword. For details on this keyword, read "FIELDS TERMINATED BY" on page 72.

```
LOAD
DATE 'MM/DD/YY'
TIME 'HH-NN-SS'
TIMESTAMP 'HH.NN.SS YY/MM/DD'
INTO TABLE SLTEST3
FIELDS TERMINATED BY ','
(
        ID,
        DT,
        TM,
        TS
)
```

PRESERVE BLANKS

The PRESERVE BLANKS keyword is used to preserve all blanks in text fields.

INTO_TABLE_PART

The *into_table_part* element is used to define the name of the table and columns that the data is inserted into.

FIELDS ENCLOSED BY

The FIELDS ENCLOSED BY clause is used to define delimiting characters around each field. The delimiter may be one character or two separate characters that precede and follow each data field in the input file. You might use one character (such as the double quote character) or a pair of characters (such as left and right parentheses) to delimit your fields. If you use the double quote mark as the delimiter and the comma as the terminator/separator, then your input might look like the following:

```
"field1", "field2"
```

If you use left and right parentheses, then your input might look like the following:

```
(field1),(field2)
```

Note that if the keyword OPTIONALLY is used, then the delimiters are optional and do not need to appear around every single piece of data.

If you specify a character value, it must be enclosed in single or double quotes. For example, the following examples have the same effect:

```
ENCLOSED BY '(' AND ')'
ENCLOSED BY "(" AND ")"
```

You can even use the single quotes to surround one enclosing character and double quotes to surround the other, for example:

```
ENCLOSED BY '(' AND ")"
```

This is potentially confusing, however, and this format is not recommended. Instead, it is recommended that you use single quotes unless you are using single quote itself as the enclosing character, for example:

```
ENCLOSED BY "'" AND "'"
```

Note that if you are using single quotes as the enclosing characters, you must double the apostrophes as shown in the clause above. For example, to produce in the database:

```
Didn't I warn you?
```

the input must be:

```
'Didn''t I warn you?'
```

Almost any printable characters may be used as the "enclosing" characters. The enclosing characters may also be specified using the hexadecimal format. For example, if a hexadecimal string is used, then the format is:

```
X 'hex_byte_string'
```

For example:

`X'3a'` means 3A hexadecimal value and specifies the colon (":")

The opening and closing characters in an enclosing pair can be identical. For example, the following is valid inside the control file:

```
ENCLOSED BY '"' AND '"'
```

If both the opening and closing characters are the same, then the ENCLOSED BY clause only needs to show the character once. For example, the following should have the same effect:

```
ENCLOSED BY '"'
ENCLOSED BY '"' AND '"'
```

When the preceding is defined in the control file, here are some examples of input and the corresponding values actually stored in the table.

```
"Hello."
Hello.

"""Ouch!"", he cried."
"Ouch!", he cried.

"""He said her last words were ""I'll never quit!""""""
"He said her last words were "I'll never quit!""

"""He said: ""Her last words were ""I'll never quit!""""""""
"He said: "Her last words were "I'll never quit!"""
```

Note that there may be enclosing characters used in the column data itself (embedded field separators). If this is the case, then you can use the TERMINATED BY clause together with the OPTIONALLY ENCLOSED BY clause to be sure the column data is enclosed correctly as described in “FIELDS TERMINATED BY” on page 72.

ENCLOSED BY input rules and examples

This section contains basic rules and examples when using enclosing characters. Each example, unless stated otherwise, contains the following control file lines:

```
FIELDS TERMINATED BY X'3a'
OPTIONALLY ENCLOSED BY "(" AND ")"
```

This means that the enclosing characters are parentheses and the separator (terminator) character is the colon — hexadecimal 3A specifies the colon (":").

- The data is to be loaded into a table with two columns, the first of which is of type VARCHAR and the second of which is type INTEGER.

Treatment of enclosed characters within the data

The ENCLOSED BY characters themselves may occur within the data. However, when occurring within the data, each of the enclosing characters should occur twice in the input for each time that it should occur once in the database.

If the input file contains:

```
(David Bowie ((born David Jones)) released 'space Oddity"):1972
```

it produces the following format in the database:

```
David Bowie (born David Jones) released 'space Oddity":1972
```

This works for deeply nested parentheses as well. If the input file contains:

```
(You((can((safely((try))this))at))home.):2
```

it produces the following value in the first column of the table.

```
You(can(safely(try)this)at)home.
```

Treatment of final enclosing character

The final enclosing character must occur an odd number of times at the end of the input. For example:

To get the following format in the database:

```
American Pie (The Day The Music Died)
```

the input file must contain:

```
(American Pie ((The Day The Music Died)))
```

Of the last three closing parentheses, the first two are treated as a single instance of the character, while the last one is treated as the enclosing character.

Embedding newline characters

When enclosing characters are used, newline characters (carriage return and/or line feed) can be embedded within a string. For example:

```
(This is a long line that can be split across two or more input
lines ((and keep the end-of-line characters)) if the enclosing
characters are used):1
```

If the field separator (the colon in the above example) is not used in the data and if there is no need to preserve newlines in the input data, then only the field separator (not the enclosing characters) is required in the input data.

If your data is fixed-width, then you do not need either the separator or the enclosing characters.

FIELDS TERMINATED BY

The FIELDS TERMINATED BY clause is used to define the separator character that distinguishes where fields end in the input file. The character must be specified in one of the following three ways:

- Surrounded by double quotes, for example, ":"

- Surrounded by single quotes, for example, ':'
- In hexadecimal format, for example, X'3A'

  When using hexadecimal format, the quotation marks must be single quotes, not double quotes.

Note that the FIELDS TERMINATED BY clause specifies a separator, not a true terminator; the specified character is not required after the last field. For example, if the colon is the separator, the following two data file formats are equivalent and valid:

```
1:2:3:
```

or

```
1:2:3
```

Note that the trailing colon is accepted, but not required, after the final field.

The OPTIONALLY ENCLOSED BY clause is used after the FIELDS TERMINATED BY clause when the character used to enclose the column data is contained in the column data itself. Following is a control file example:

```
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY "'"
```

In the example above, the separator is a comma.

The single quote is defined as the character that encloses embedded field separators (commas) in the data file. Note that the OPTIONALLY ENCLOSED BY clause may use either single or double quotes to delimit the enclosing characters. The following example:

```
OPTIONALLY ENCLOSED BY '('AND")"
```

illustrates the use of both single and double quotes for *enclose_char* in the syntax:

```
ENCLOSED BY enclose_char [AND enclose_char]
```

The example is unusual, but its potential for confusion makes it worth noting.

The following example summarizes the use of separators and enclosing characters. In this example, the ":" (colon) is defined as the separator (FIELDS TERMINATED BY) and the parentheses are used to enclose the ":" (colon), which is embedded in the field and should not be interpreted as a separator. The example also contains two fields, the first of which is VARCHAR and the second of which is INTEGER.

Data File Example

```
(This colon : is enclosed by parentheses and is not a separator):12345
```

Control File Example

```
LOAD DATA
CHARACTERSET MSWINDOWS
INFILE 'test6.dat'
INTO TABLE SLTEST
FIELDS TERMINATED BY X'3a' -- X'3a' == ':'
OPTIONALLY ENCLOSED BY '(' AND ")"
(
    TEXT,
    ID
)
```

POSITION

The POSITION keyword is used to define a field's position in the logical record. Both the start and the end position must be defined.

NULLIF

The NULLIF keyword is used to give a column a NULL value if the appropriate field has a specified value. An additional keyword specifies the value the field must have. The keyword BLANKS sets a NULL value if the field is empty; the keyword NULL sets a NULL value if the field is the string 'NULL'; the definition '*string*' sets a NULL value if the field matches the string '*string*'; the definition '((start : end) = '*string*')' sets a NULL value if a specified part of the field matches the string '*string*'.

Using NULLIF keyword with keyword BLANKS

The following example shows the use of the NULLIF keyword with the keyword BLANKS to set a NULL value if the field is empty. It also shows the use of the keyword NULL to set a NULL value if the field is the string 'NULL'.

```
LOAD
INFILE 'test7.dat'
INTO TABLE SLTEST
FIELDS TERMINATED BY ','
(
        NAME    VARCHAR NULLIF BLANKS,
        ADDRESS VARCHAR NULLIF NULL,
        ID      INTEGER NULLIF BLANKS
)
```

Using NULLIF keyword with keyword BLANKS

The following example uses the definition '((start : end) = '*string*')' for the third field in the input file. This syntax only works with fixed-width fields because the exact position of the 'string' must be specified.

```
LOAD
INFILE '7b.dat'
INTO TABLE t7
(
    NAME CHAR(10) POSITION(1:10) NULLIF BLANKS,
    ADDRESS CHAR(10) POSITION(11:20) NULLIF NULL,
    ADDR2 CHAR(10) POSITION(21:30) NULLIF((21:30)='MAKEMENULL')
)
```

Note that in this example, the string is case sensitive. 'MAKEMENULL' and 'makemenull' are not equivalent.

Loading fixed-format records

Control File Example 1

```
-- EXAMPLE 1 uses multiple columns in fixed-width field

OPTIONS(ARRAYSIZE=3)

LOAD
INFILE 'test1.dat'
INTO TABLE SLTEST
(
   "NAME"  POSITION(1-5),
   ADDRESS POSITION(6:10),
   ID      POSITION(11-15)
)
```

Control File Example 2

```
-- EXAMPLE 2
OPTIONS (SKIP = 10, ERRORS = 5)
-- Skip the first ten records. Stop if
-- errorcount reaches five.
LOAD DATA
INFILE 'sample.dat'
-- import file is named sample.dat
INTO TABLE TEST1 (
IDINTEGER POSITION(1-5),
ANOTHER_ID INTEGER POSITION(8-15),
DATE1      POSITION(20:29) DATE 'YYYY-MM-DD',
DATE2      POSITION(40:49) DATE 'YYYY-MM-DD' NULLIF NULL)
```

Loading variable-length records

This section contains examples of the control file when loading data from a variable-length import file:

Control File Example 3

```
-- EXAMPLE 1 uses multiple columns that have separators rather than
-- fixed length fields.

    LOAD
    INFILE 'test1.dat'
    INTO TABLE SLTEST
    FIELDS TERMINATED BY ','
    (
       NAME,
       ADDRESS,
       ID
    )
```

Control File Example 4

```
LOAD DATA
INFILE 'EXAMP2.DAT'
INTO TABLE SUPPLIERS
FIELDS TERMINATED BY ','
(NAME VARCHAR, ADDRESS VARCHAR, ID INTEGER)
-- EXAMPLE 2
OPTIONS (SKIP=10, ERRORS=5)
-- Skip the first ten records. Stop if
-- errorcount reaches five.
LOAD
DATE 'YYYY-MM-DD HH:NN:SS'
-- The date format in the import file
INFILE 'sample.dat'
-- The import file
INTO TABLE TEST1
-- data is inserted into table named TEST1
FIELDS TERMINATED BY X'2C'
-- Field terminator is HEX ',' == 2C
-- This line could also be:
-- FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '[' AND ')'
-- Fields may be enclosed
-- with '[' and ')'
(
ID INTEGER,
ANOTHER_ID DECIMAL(2),
DATE1 DATE(20) DATE 'YYYY-MM-DD HH:NN:SS',
DATE2 NULLIF NULL
)
-- ID is inserted as integer
-- ANOTHER_ID is a decimal number with 2
```

```
-- digits.
-- DATE1 is inserted using the datestring
-- given above
-- The default datestring is used for DATE2.
-- If the column for DATE2 is 'NULL' a NULL is
-- inserted.
```

Running a sample load using solidDB Speed Loader (solload)

Note that the files that are referred to in this section are contained in the `samples/importexport/solload` directory.

1. Start solidDB.
2. Create the table by using the `load.sql` script and your solidDB SQL Editor.
3. Start loading by entering the command below:

   solload "tcpip 1964" dba dba delim.ctr

   The user name and password are assumed to be 'dba'. To use the fixed length control file, enter the command below:

   solload "tcpip 1964" dba dba fixed.ctr

   The output of a successful loading using `delim.ctr` or `fixed.ctr` is:

   ```
   IBM solidDB Speed Loader v.06.30.0015
   (c) Solid Information Technology Ltd. 1993, 2008
   Load completed successfully, 19 rows loaded.
   ```

Hints to speed up loading

The following hints can be used to ensure that loading is done with maximum performance:

- Connect locally if possible; it is faster not to load data over the network.
- Increase the number of records committed in one batch. By default, commit is done after each record.
- Disable transaction logging.

You must use the **LogEnabled** parameter to disable logging. The following lines in the `solid.ini` file will disable logging:

```
[Logging]
LogEnabled=no
```

After the loading has been completed, remember to enable logging again. The following line in the `solid.ini` file will enable logging:

```
[Logging]
LogEnabled=yes
```

Note: Running the server in production use with logging disabled is strongly discouraged. If logs are not written, no recovery can be made if an error occurs due to power failure, disk error etc.

solidDB Export (solexp)

solidDB Export (solexp) is a tool for unloading data from a solidDB database to ASCII files. solidDB Export produces both the import file, that is, the file containing the exported ASCII data, and the control file that specifies the format of the import file. solidDB Speed Loader can directly use these files to load data into a solidDB database.

Note:

The user name used for performing the export operation must have select rights on the table exported. Otherwise no data is exported.

Starting solidDB Export

Start solidDB Export with the command `solexp`. If you start solidDB Export with no arguments, a summary of the arguments with a brief description is displayed.

The syntax for the `solexp` command is:

`solexp [options] [servername] username [password] {tablename | *}`

where

- *`options`* is optional; the argument can be:

Table 21. solexp command options

Option Syntax	Description
-c *dir*	Change working directory
-e *sql_string*	Execute SQL string for export.
-f *filename*	Execute SQL string from file for export.
-l *filename*	Write log entries to this file.
-L *filename*	Append log entries to this file.
-o *filename*	Write exported data to this file.
-s *schema_name*	Use only this schema for export.
-C *catalog_name*	Set the default catalog from where data is read from or written to.
-p	Preserve case of schema and table names.
-8	Output 8-bit names to .crt file (disables UNICODE names).
-h, -?	Help = Usage.
-x pwdfile: *filename*	Read password from the file.

- *`servername`* is optional; it defines the network name of a solidDB that you are connected to. Logical Data Source Names can also be used with tools; refer to 6, “Managing network connections,” on page 99 further information. The given network name must be enclosed in double quotes.

  Note: If you do not specify *`servername`*, solexp uses the environment-dependent defaults for the connection (`NmPipe solid` in Windows; `UPipe solid` in Linux and UNIX).
- *`username`* is mandatory; it identifies the user and determines the user's authorization. Without appropriate rights, execution is denied.
- *`password`* is
  - mandatory, if the password is not read from a file (defined with option -x pwdfile: *filename*)

– optional, if the password is read from a file

- *tablename* or * is mandatory. The symbol * can be used to export all tables with one command. However, it cannot be used as a wildcard.

  Note: The -t *tablename* (Export table) option is still supported in order to keep old scripts valid.

Example

```
solexp -CMyCatalog -sMySchema -ofile.dat "tcp 1315" MyID My_pwd MyTable
```

Troubleshooting

- When there is an error in the command line entry, solexp gives you a list of the possible options as a result. Check your entries on the command line.
- Username, password and table name are always expected:

  For example, with the command

  ```
  solexp "tcp 1315" dba dba
  ```

  you may receive a SOLID Communication Error 21306. This is because there was no server listening to the environment-dependent default. In this case, solexp assumes:

  – "tcp 1315" is the username

  – dba is the password

  – dba is the table name

  In this case, the correct command is, for example:

  ```
  solexp "tcp 1315" dba dba myTable
  ```

- If you omit the name of the schema, you may get a message saying that the specified table could not be found. The solexp program cannot find the table if it does not know which schema to look in.

solidDB Data Dictionary (soldd)

solidDB Data Dictionary (soldd) is a tool for retrieving data definition statements from a solidDB database. solidDB Data Dictionary produces a SQL script that contains data definition statements describing the structure of the database. The generated script contains definitions for tables, views, indexes, triggers, procedures, sequences, publications, and events.

Note:

1. User and role definitions are not listed for security reasons.
2. The user name used for performing the export operation must have select right on the tables. Otherwise the connection is refused.

Starting solidDB Data Dictionary

Start solidDB Data Dictionary with the command soldd. If you invoke solidDB Data Dictionary with no arguments, you'll see a summary of the arguments with a brief description. The command line syntax is:

```
soldd options servername username password [ tablename]
```

where options can be:

Table 22. soldd command options

Option Syntax	Description
-c *dir*	Change working directory
-m	Expect input in multi-byte character format.
-o *filename*	Write data definitions to this file.
-O *filename*	Append data definitions to this file.
-C *catalog_name*	Set the default catalog from where data definitions are read from or written to.
-s *schema_name*	List definitions from this schema only.
-p	Preserve case of schema and table names.
-8	Output 8-bit names to .crt file (disables UNICODE names).
-h, -?	Help = Usage.
-x tableonly	List table definitions only.
-x indexonly	List index definitions only.
-x viewonly	List view definitions only.
-x sequenceonly	List sequence definitions only.
-x procedureonly	List procedure definitions only.
-x publicationonly	List publication definitions only.
-x eventonly	List event definitions only.
-x triggeronly	List trigger definitions only.
-x schemaonly	List schema definitions only.
-x hiddennames	List internal constraint names only.
-x pwdfile: *filename*	Read password from the file.

Servername is the network name of a solidDB server that you are connected to. Logical Data Source Names can also be used with tools; Refer to 6, “Managing network connections,” on page 99 for further information. The given network name must be enclosed in quotes.

Username is required to identify the user and to determine the user's authorization. Without appropriate rights, execution is denied.

Password is the user's password for accessing the database.

When there is an error in the command line, the solidDB Data Dictionary gives you a list of the possible options as a result. Please be sure to check the command line you entered.

solidDB Data Dictionary Examples

```
soldd -odatabase.sql "tcp database_server 1313" dbadmin f1q32j4
```

Print the definition of procedure TEST_PROC:

```
soldd -x procedureonly " " dba dba TEST_PROC
```

Note:

1. If no table name is given, all definitions to which the user has rights are listed.
2. If the *objectname* parameter is provided with one of the -x options, the name is used to print only the definition of the named object.
3. The -t *tablename* option is still supported in order to keep old scripts valid.

Tools sample: reloading a database

This example demonstrates how a solidDB database can be reloaded to a new one. At the same time the use of each solidDB tool is introduced with an example. Note that delete and update operations can leave gaps (unused space) in the database. The reload is a useful procedure since it will rewrite the database without gaps and shrink the size of the database file `solid.db` to a minimum.

To reload the database

1. Extract data definitions from the old database.
2. Extract data from the old database.
3. Replace the old database with a new one.
4. Load data definitions into a new database.
5. Load data into the new database.

Reloading the database: Walkthrough

In this example, the server name is solidDB and the protocol used for connections is TCP/IP, using port 1964. Therefore, the network name is "tcpip 1964". The database has been created with the user name "dbadmin" and the password "password".

1. Data definitions are extracted with solidDB Data Dictionary. Use the following command line to extract a SQL script containing definitions for all tables, views, triggers, indexes, procedures, sequences, and events. The default for the extracted SQL file is `soldd.sql`.

   soldd "tcpip 1964" dbadmin password

   With this command, all data definitions are listed into one file, `soldd.sql` (the default name). As mentioned earlier, user and role definitions are not listed for security reasons. If the database contains users or roles, they must be appended into this file.
2. All data is extracted with solidDB Export. The export results in control files (files with the extension .ctr) and data files (files with the extension .dat). The default file name is the same as the exported table name. In 16-bit environments, file names longer than eight letters are concatenated. Use the following command line to extract the control and data files for all tables.

   solexp "tcpip 1964" dbadmin password *

With this command data is exported from all tables. Each table's data is written to an import file named `table_name.dat`. A separate control file `table_name.ctr` is written for each table name.

3. A new database can be created to replace the old one by deleting the `solid.db` and all `sol#####.log` files from the appropriate directories. When solidDB is started for the first time after this, a new database is created.

   Note: It is recommended that a backup is created of the old database before it is deleted. This can be done using solidDB Remote Control.

4. Use the following command line to create a backup using solidDB Remote Control:

   solcon -eBACKUP "tcpip 1964" dbadmin password

   With this command, a backup is created. The option -e precedes an administration command.

5. Load data definitions into the new database. This can be done using solidDB SQL Editor. Use the following command line to execute the SQL script created by solidDB Data Dictionary.

   solsql -fSOLDD.SQL "tcpip 1964" dbadmin password

   With this command, data definitions are loaded into the new, empty database. Definitions are retrieved with the option -f from the file `soldd.sql`. Connection parameters are the same as in the earlier examples.

   The previous two steps can be performed together by starting solidDB with the following command line. The option -x creates a new database, executes commands from a file, and exits. User name and password are defined as well.

   solid -Udbadmin -Ppassword -x execute:soldd.sql

6. Load data into the new database. This is done with solidDB Speed Loader. To load several tables into the database, a batch file containing a separate command line for each table is recommended. In Unix-based operating systems, using the wildcard symbol * is possible. Use the following command line to load data into the new database.

   solload "tcpip 1964" dbadmin password table_name.ctr

7. With this command, data for one table is loaded. The server is online.

   Batch files that can be used are:

   - Shell scripts in Unix environments
   - `.com` scripts in VMS
   - `.bat` scripts in Windows

5 Performance tuning

This chapter discusses techniques that you can use to improve the performance of solidDB. The topics included in this chapter are:

- Logging and Transaction Durability
- Choosing isolation levels
- Controlling memory consumption
- Tuning network messages
- Tuning I/O
- Tuning checkpoints
- Reducing Bonsai Tree size by committing read-only transactions
- Diagnosing poor performance

For tips on optimizing advanced replication, see *IBM solidDB Advanced Replication User Guide*.

Tip: The following parameters help you improve database performance or balance performance against safety. These parameters are discussed in more detail in Appendix A, "Server-side configuration parameters," on page 117. The DurabilityLevel parameter is also discussed in 5, "Performance tuning."

- **IsolationLevel**
- **DurabilityLevel**
- **DefaultStoreIsMemory**

Logging and transaction durability

This chapter discusses transaction durability from a theoretical perspective. For more information on choosing the transaction durability level and setting it, refer to *IBM solidDB SQL Guide*.

Background

When a transaction is committed, the database server writes data to two locations: the database file, and the transaction log file. However, the data is not necessarily written to those two locations at the same time. When a transaction is committed, the server normally writes the data to the transaction log file immediately — that is, as soon as the server commits the transaction. The server does not necessarily write the data to the database file immediately. The server may wait until it is less busy, or until it has accumulated multiple changes, before writing the data to the database file.

If the server shuts down abnormally (due to a power failure, for example) before all data has been written to the database file, the server can recover 100% of committed data by reading the combination of the database file and the transaction log file. Any changes since the last write to the database file are in the transaction log file. The server can read those changes from the log file and then use that information to update the database file. The process of reading changes from the log file and updating the database file is called "recovery". At the end of the recovery process, the database file is 100% up to date.

The recovery process is automatically executed always when the server restarts after an abnormal shutdown. The process is generally invisible to the user (except that there may be a delay before the server is ready to respond to new requests).

Not surprisingly, to have 100% recovery, you must have 100% of the transactions written to the log file. Normally, the database server writes data to the log file at the same time that the server commits the data. Thus committed transactions are stored on disk and will not be lost if the computer is shut down abnormally. This is called "strict durability". The data that has been committed is "durable", even if the server is shut down abnormally.

If durability is 'strict", the user is not told that his data has been committed until AFTER that data was successfully written to the transaction log on disk — this ensures that the data is recoverable if the server shuts down abnormally. Strict durability makes it almost impossible to lose committed data unless the hard disk drive itself fails.

If durability is "relaxed", the user may be told that the data has been committed even before the data has been written to the transaction log on disk. The server may choose to delay writing the data, for example, by waiting until there are several transactions to write. If durability is relaxed, the server may lose a few committed transactions if there is a power failure before the data is written to disk.

solidDB allows to control the durability level in variety of ways. For the server-wide setting, the parameter **DurabilityLevel** in the [Logging] section may take three values: 3 (for 'strict"), 1 (for "relaxed") and 2 (for "adaptive").

Adaptive durability is meant for HotStandby operation. If durability is "adaptive", then the server follows the rules below:

- If the server is a Primary server in a HotStandby system, and if the Secondary is active, then the server (Primary server) uses relaxed durability;
- In all other situations, the server uses strict durability.

Note:

- The above behavior is observed only if the value of the [HotStandby] parameter **SafenessLevel** is set to 2safe (default). If this parameter is set to any other value, the server uses relaxed durability in all cases.
- If HotStandby is not enabled, the "adaptive" setting is treated as 'strict".

Balancing performance and safety

Historically, the goal of most database servers has been to maximize safety, that is, to make sure that data is not lost due to a power failure or other problems. These database servers use 'strict durability". This approach is appropriate for many types of data, such as accounting data, where it is often unacceptable to lose track of even a single transaction.

Some database servers have been designed to maximize performance, without regard to safety. This is acceptable in situations where, for example, you only need to sample data, or where the server can simply operate on the most recent set of data, regardless of the size of that set. As an example, suppose that you have a server that contains statistical data about performance — e.g. which computers experience the heaviest loads at particular times of the day. You might use such information to balance the load on your computers. This information changes over time, and "old" data is less valuable than "new" data. In fact, you might completely

discard any data that is more than a week old. If you were to lose the performance and load balancing data, then your system would still function, and within a week you would have acquired a complete set of new data (assuming that you normally discard data older than one week). In this situation, occasional or small data loss is acceptable, and performance may be more important.

solidDB allows you to specify whether you want logging to be 'strict" to guarantee that all committed data can be recovered after an unexpected shutdown, or "relaxed" to allow some recent transactions to be lost in some circumstances.

How relaxed transaction durability can improve performance

You can increase performance by telling the server that it does not necessarily have to write to the log file at the same time that it commits data. This allows the server to write to the log file later, perhaps when the server is less busy, or when several transactions can be written at once. This is called " relaxed durability". It increases performance by decreasing the I/O (Input/Output) load.

If you set the transaction durability level to "relaxed", then you risk losing some data if the server shuts down abnormally after it has committed some data but before it has written that data to the transaction log. Therefore, you should use relaxed durability ONLY when you can afford to lose a small amount of recent data.

Standards compliance

Transaction durability is not part of the ANSI standard for SQL-99.

Limitations on transaction durability

CAUTION:

When you use "relaxed" transaction durability, you risk losing data. If the database server shuts down abnormally (due to a power failure, for example), the server will lose any committed transactions that were not written to the transaction log file. If you use relaxed durability, some transactions may not have been written to the log file yet, even though those transactions were committed. You should ONLY use relaxed durability when you can afford to occasionally lose a small amount of the most recent data.

If you want to set a maximum delay time before the server writes data, set the **RelaxedMaxDelay** parameter in the `solid.ini` configuration file. For more information about the **RelaxedMaxDelay** parameter, see the section “Logging section” on page 142.

Choosing transaction isolation levels

Concurrency control is based on an application's requirements. Some applications need to execute as if they had exclusive ownership of the database. Other applications can tolerate some degree of interference from other applications running simultaneously. To meet the needs of different applications, the SQL-92 standard defines four levels of isolation for transactions. By principle, solidDB cannot read uncommitted data. The reason is that it sacrifices the consistent view and potentially also database integrity. The three supported isolation levels are explained below.

- Read Committed

This isolation level allows a transaction to read only committed data. Nonetheless, the view of the database may change in the middle of a transaction when other transactions commit their changes.

- Repeatable Read

  This isolation level allows a transaction to read only committed data and guarantees that read data will not change until the transaction terminates. solidDB additionally ensures that the transaction sees a consistent view of the database. When using optimistic concurrency control, conflicts between transactions are detected by using transaction write-set validation. This means that the server validates only write operations, not read operations. For example, if a transaction involves one read and one update, solidDB validates that no one has updated the same row in between the read operation and the update operation. In this way, lost updates are detected, but the read is not validated. With transaction write-set validation, phantom updates may occur and transactions are not serializable.

- Serializable

  This isolation level allows a transaction to read only committed data with a consistent view of the database. Additionally, no other transaction may change the values read by the transaction before it is committed because otherwise the execution of transactions cannot be serialized in the general case.

  solidDB can provide serializable transactions by detecting conflicts between transactions. It does this by using both write-set and read-set validations. Because no locks are used, all concurrency control anomalies are avoided, including the phantom updates. This feature is enabled by using the command SET TRANSACTION ISOLATION LEVEL SERIALIZABLE, which is described in *Appendix: "solidDB SQL Syntax"* in *IBM solidDB SQL Guide*.

Note: The SERIALIZABLE isolation level is available for disk-based tables only.

Setting the isolation level

To set the isolation level, use one of the following SQL commands:

```
SET ISOLATION LEVEL
    {READ COMMITTED | REPEATABLE READ | SERIALIZABLE}
SET TRANSACTION ISOLATION LEVEL
    {READ COMMITTED | REPEATABLE READ | SERIALIZABLE}
```

For example:

```
SET ISOLATION LEVEL REPEATABLE READ;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
```

Note that solidDB supports both "transaction-level" and 'session-level" isolation level commands. For more details, see the descriptions in *IBM solidDB SQL Guide*, *Appendix: solidDB SQL syntax*.

Controlling memory consumption

Main memory is allocated dynamically according to system usage and the operating system environment. The basic element of the memory management system is a pool of central memory buffers of equal size. You can configure the amount and size of memory buffers to meet the demands of different application environments.

Note:

Right after the solidDB startup, Microsoft Windows reports a significantly smaller process size than the real allocated size is. This is because cache pages are allocated at this stage, but Microsoft Windows excludes them from the process size until they are used for the first time. As opposed to Microsoft Windows, Unix based operating systems include the cache pages and report a bigger process size.

Controlling process size

The process size, as such, does not directly correspond to the actual database memory consumption, because the process size contains also non-database elements. The process size includes elements as follows:

- The cache size. The `solid.ini` default value is 32 Mbytes.
- The code footprint is approximately three Mbytes, but it initialises different libraries and can grow up to 8 Mbytes.
- Client threads. Each client consumes a few hundred kilobytes of main memory.
- Dynamic memory reserved for command handling. This memory allocation deals with execution plans, temporary data, and so on.
- Statement cache. When solidDB executes SQL statements, it parses and optimizes them first. This can be time consuming. The server can store the parsed and optimised statements in the virtual memory. This is called the statement cache.
- The hash table for the transaction lookup table.
- Transaction and sort buffers.
- The **LockHashSize** parameter affects the memory consumption. This parameter defines the number of elements in the lock hash table.
- The accessed tables are also buffered in the main memory.

The elements above are the main elements affecting the process size.

You can control the process size by using the admin command and parameters presented in the following sections. The violations of process limits are logged in the `solmsg.out` log file.

ADMIN COMMAND 'info processsize';

The ADMIN COMMAND 'info processsize'; command returns the current amount of memory that the in-memory database process uses. The value returned is a VARCHAR, and it indicates the number of kilobytes used by the process. Note that this returns the amount of virtual memory used, not the amount of physical memory used.

ProcessMemoryLimit parameter

The **ProcessMemoryLimit** parameter specifies the maximum amount of virtual memory that can be allocated to the in-memory database process. The **ProcessMemoryLimit** parameter is controlled with the **ProcessMemoryCheckInterval** parameter.

If the **ProcessMemoryCheckInterval** parameter value is 0 (factory value), the **ProcessMemoryLimit** parameter is not effective, that is, there is no process memory limit.

The factory value for **ProcessMemoryLimit** is 1G (one Gigabyte). Set the parameter to a value that will ensure that the in-memory database process will fit entirely within physical memory. The following factors impact the amount of memory needed:

- the amount of physical memory in the computer
- the amount of memory used by the operating system
- the amount of memory used by in-memory tables (including Temporary Tables, Transient Tables, and "normal" in-memory tables) and the indexes on those in-memory tables
- the amount of memory set aside for the solidDB server's cache (the **CacheSize** `solid.ini` configuration parameter)
- the amount of memory required by the connections, transactions and statements running concurrently in the server. The more concurrent connections and active statements there are in the server, the more working memory the server requires. Typically, you should allocate at least 0.5 MB of memory for each client connection in the server.
- the memory used by other processes (programs and data) that are running in the computer

When the limit is reached, that is, when the in-memory database process uses up 100% of the memory specified by **ProcessMemoryLimit**, the server will accept admin commands only. You can use the **ProcessMemoryWarningPercentage** and **ProcessMemoryLowPercentage** parameters to warn you about increasing memory consumption.

ProcessMemoryLowPercentage parameter

The **ProcessMemoryLowPercentage** parameter sets a warning limit for the total process size. The limit is expressed as percentage of the **ProcessMemoryLimit** parameter value. Prior to exceeding this limit, you have exceeded the warning limit defined by using the **ProcessMemoryWarningPercentage** parameter and received a warning. When the **ProcessMemoryLowPercentage** limit is exceeded, a system event is given.

The limit set with **ProcessMemoryLowPercentage** must be higher than the **ProcessMemoryWarningPercentage** limit. For example, if the **ProcessMemoryWarningPercentage** is set to 82, the **ProcessMemoryLowPercentage** value must be at least 83.

ProcessMemoryWarningPercentage parameter

The **ProcessMemoryWarningPercentage** parameter sets the first warning limit for the total process size. The warning limit is expressed as percentage of the **ProcessMemoryLimit** parameter value. When the **ProcessMemoryWarningPercentage** limit is exceeded, a system event is given.

The limit set with **ProcessMemoryWarningPercentage** must be lower than the **ProcessMemoryLowPercentage** limit.

ProcessMemoryCheckInterval parameter

The process size limits are checked periodically. The check interval is set with the **ProcessMemoryCheckInterval** parameter. The interval is given in milliseconds.

The minimum non-zero value is 1000 (ms). Only values 0 or 1000 or above 1000 (1 second) are allowed. If the given value is above 0 but below 1000, an error message is given.

The factory value is 0, that is, process size checking is disabled.

The **ProcessMemoryCheckInterval** also controls the **ProcessMemoryLimit** parameter; if the **ProcessMemoryCheckInterval** parameter value is 0, the **ProcessMemoryLimit** parameter is not effective, that is, there is no process memory limit.

Tuning your operating system

Your operating system may store information in:

- real (physical) memory
- virtual memory
- expanded storage
- disk

Your operating system may also move information from one location to another. Depending on your operating system, this movement is called paging or swapping. Many operating systems page and swap to accommodate large amounts of information that do not fit into real memory. However, this takes time. Excessive paging or swapping can reduce the performance of your operating system and indicates that your system's total memory may not be large enough to hold everything for which you have allocated memory. You should either increase the amount of total memory or decrease the amount of database cache memory allocated.

Database cache

The information managed by solidDB is stored either in memory or on disk. Since memory access is faster than disk access, it is desirable for data requests to be satisfied by access to memory rather than access to disk.

Defining database cache size

Database cache uses available memory to store information that is read from the hard disk, in a disk-based database. It is also used to buffer the database pages while the server is executing the checkpoint -- equally in a disk based and an in-memory database. When an application next time requests this information, the data is read from memory instead of from the hard disk. The default value of cache depends on the platform used and can be changed through the **CacheSize** parameter. Increasing the value is recommended when there are several concurrent users.

If a database is primarily disk-based, the following estimates can be used:

- 0.5 MB per each concurrent user of the system

or

- 2-5% of the database size,

When estimating the necessary cache size by using the values above, use the larger value. If the database is purely an in-memory database, the factory value will suffice. When decreasing the cache size, note that in order to facilitate efficient checkpoint activity, the size should not be less than 8 MB.

You should increase the value of **CacheSize** carefully. If the value is too large, it leads to poor performance because the server process does not fit completely in memory and therefore swapping of the server code itself occurs. If, on the other hand, the cache size is too small, the cache hit rate remains poor. The symptoms of poor cache performance are database queries that seem to be slower than expected and excessive disk activity during queries.

You can verify if the server is retrieving most of the data from disk instead of from RAM by checking the cache hit rate using the command `ADMIN COMMAND 'status'` or by checking the overall cache and file ratio statistics using `ADMIN COMMAND 'perfmon'`. For details on these commands, read "Performance counters (perfmon)" on page 19 and "Checking overall database status" on page 18. Note that the cache hit rate should be better than 95%.

Dynamically changing database cache size

You can change the **CacheSize** value dynamically as follows:

admin command 'parameter IndexFile.CacheSize=40mb'

Note:

The cache size cannot be decreased.

solidDB uses a hash table to ease access to the cache. The hash table size equals the number of pages in the cache. This guarantees almost collision-free access. If the cache size is increased dynamically, the hash table is not automatically enlarged. This results in a higher collision probability. To avoid this, you can use the **ReferenceCacheSizeForHash** parameter to accommodate the enlarged cache. The **ReferenceCacheSizeForHash** parameter value is used for calculating the cache hash table size. You should only use the parameter if you know, in advance, what will be the maximum cache size during the server lifecycle. On the other hand, if the value is not given, hash table collisions may occur when the cache size is increased.

Note:

The **ReferenceCacheSizeForHash** parameter value must not be smaller than the **CacheSize** value. If it is, the **ReferenceCacheSizeForHash** parameter value is rejected and the default value is used. Also, a message is printed to the `solmsg.out` log file.

Sorting

By default, solidDB does all sorting in memory. The amount of memory used for sorting is determined by the parameter **SortArraySize** in the [SQL] section. If the amount of data to be sorted does not fit into the allocated memory, you may want to increase the value of the parameter **SortArraySize**.

Note that it may seem that the correct setting for the size of the sort array must accommodate the largest expected result set (that cannot be ordered by key values); however, there are some non-intuitive consequences to consider when increasing the sort array size.

If increasing the value of the **SortArraySize** results in slower, rather than faster query times, then it is likely that one of the following behaviors of the Optimizer is involved:

- The **SortArraySize** parameter affects whether indices are used for sorting. If the **SortArraySize** setting is large, the Optimizer is likely to use the sort array for sorting, rather than using the available indices for sorting. If the **SortArraySize** is small, the Optimizer is likely to use the available indices for sorting. In some cases (especially those with small result sets), a small **SortArraySize** setting performs better than a large **SortArraySize** setting.

- The **SortArraySize** parameter affects the way that the Optimizer performs GROUP operations. The Optimizer considers a GROUP operation on non-sorted result sets as an expensive operation. Thus, with smaller settings for the SortArraySize, the optimizer causes the result sets to be sorted before performing the GROUP operation. With larger settings for the **SortArraySize**, the GROUP operation tends to proceed without first sorting the result set. In some cases, this can result in slower performance for the larger settings of the **SortArraySize** than for the smaller settings.

Note that for large sorts, or when there is not enough memory to increase the value of **SortArraySize**, you should activate the external sort, which stores intermediate information to disk.

The external disk sort is activated by adding the following section and parameters in the configuration file `solid.ini`:

```
[sorter]
TmpDir_1 = c:\tmp
```

Additional sort directories are added with similar definitions:

```
[sorter]
TmpDir_1 = c:\tmp
TmpDir_2 = d:\tmp
TmpDir_3 = e:\tmp
```

Defining more than one sorter temporary directory on separate physical disks may significantly improve sort performance by balancing the I/O load to multiple disks.

Optimized sorts

Some queries implicitly require sorting. For example, if the SQL Optimizer chooses a JOIN operation to use the MERGE JOIN algorithm, the result sets to be joined require sorting before the join can occur. You can query the Optimizer's decisions from solidDB using the EXPLAIN PLAN FOR statement. For details, read the description of the EXPLAIN PLAN FOR command in *IBM solidDB SQL Guide*.

Sorting occurs only if the result set is not returned automatically in the correct order. If the table data is accessed using the primary key or index, then the result set is automatically in the order specified by the index in use. Hence, you can significantly improve server performance by designing primary keys and indices to support the ordering requirements of frequently used, performance-critical queries.

Using in-memory database

The solidDB database products use two integrated database engines: one is a traditional disk-based engine and the other is a main memory engine allowing to create tables that reside permanently in main memory. Also the indexes created for those tables are stored totally in main memory. When using the in-memory database capability you may choose, for each table, which is the storage for the table: disk or memory. A solidDB server process running in-memory tables is significantly larger than a purely disk-based server process. To evaluate the amount of memory required by the in-memory tables and their indexes, refer to *IBM solidDB In-Memory Database User Guide*.

Tuning network messages

You can improve solidDB performance in reading large result sets by instructing a solidDB server to return several result set rows in one network message. To activate this functionality, you edit one or both of the following parameters in the [Srv] section of the solidDB server's `solid.ini` configuration file.

- **RowsPerMessage**: The default value is 10.
- **ExecRowsPerMessage**: The default value is 2.

For more information about these two parameters, see Appendix A, "Server-side configuration parameters," on page 117.

Tuning I/O

The performance of many software systems is inherently limited by disk I/O. Often CPU activity must be suspended while I/O activity completes.

Distributing I/O

Disk contention occurs when multiple processes try to access the same disk simultaneously. To avoid this, move files from heavily accessed disks to less active disks until they all have roughly the same amount of I/O.

Follow these guidelines:

- Use a separate disk for log files.
- Divide your database into several files and place each of these database files on a separate disk. Read "Managing database files and caching (IndexFile section)" on page 50.
- Consider using a separate disk for the external sorter

It is usually faster to scan a table if the disk file is contiguous on the disk, rather than spread across many non-contiguous disk blocks. To reduce existing fragmentation, you may want to run defragmentation software if one is available on your system. If your database file is growing, you may be able to reduce future file fragmentation by using the configuration parameter **ExtendIncrement**. Increasing the size of this parameter tells the server to allocate larger amounts of disk space when it runs out of space. (Note that this does not guarantee contiguity because the operating system itself may allocate non-contiguous sectors to satisfy even a single request for more space.) As a general rule, larger values of **ExtendIncrement** improve performance slightly, while smaller values keep the database size slightly smaller. See Appendix A, "Server-side configuration parameters," on page 117, for more details about **ExtendIncrement**.

Setting the MergeInterval parameter

solidDB's indexing system consists of two storage structures:

- the Bonsai Tree, which stores new data in central memory, and
- the main storage tree, which stores more stable data.

As the Bonsai Tree performs concurrency control, storing delete, insert, and update operations, as well as key values, it merges new committed data to the storage tree as a highly-optimized batch insert. This offers significant I/O optimization and load balancing.

You can adjust the number of index inserts made in the database that causes the merge process to start by setting the following parameter in the General section of the `solid.ini` file. For example:

```
MergeInterval = 1000
```

Normally the recommended setting is the default value, which is cache size dependent. The default is calculated dynamically from the cache size, so that only part of the cache is used for the Bonsai Tree. If you change the merge interval, be sure that the cache is large enough to accommodate the Bonsai Tree. The longer the merge interval is (i.e. the more data that is stored in memory before being moved to the main storage tree), the larger the cache needs to be.

Note: If the merge interval setting is too big to allow the Bonsai Tree to fit into cache, then it is flushed partially to the disk; this has an adverse affect on performance. Hence, avoid setting merge intervals that are too large. On a diskless system, the Bonsai Tree will fill the available memory and the Diskless server will run out of memory.

Note: Although the server will have higher performance if merge intervals are less frequent (i.e. batch inserts are larger), you may also see less consistent response times. If your highest priority is not overall throughput, but is instead to minimize the longest response time, then you may want to make merge intervals more frequent rather than less frequent. More frequent merges will reduce the worst case delays that interactive users may experience.

For details on detecting and preventing performance problems associated with Bonsai Tree growth, read "Reducing Bonsai Tree size by committing transactions" on page 94.

Tuning checkpoints

Checkpoints are used to store a transactionally-consistent state of the database quickly onto the disk.

Checkpoints affect:

- runtime performance
- recovery time performance

Checkpoints cause solidDB to perform data I/O with high priority, which momentarily reduces the run-time performance. This overhead is usually small. As with merge intervals, less frequent checkpoints may mean less frequent, but longer, delays before the system responds to interactive queries. More frequent checkpoints tend to minimize the worst case delays that an interactive user might experience. However, such delays may be more frequent even if they are shorter.

It is possible to control the execution of checkpoints to prevent them from occurring during, for example, periods of high user volume. You may:

- Set configuration parameters in the `solid.ini` file.
  - Set the **CheckpointInterval** parameter in the `solid.ini` configuration file. The default checkpoint interval is every 50000 log writes.
  - Set the **MinCheckpointTime** parameter in `solid.ini`.

  For more information about these parameters, see Appendix A, "Server-side configuration parameters," on page 117. To learn how to change a parameter value, see "Managing server-side parameters" on page 55 in this guide.

- Force a checkpoint by using the makecp command. For details on makecp, read "Creating checkpoints" on page 37.

Frequent checkpoints can reduce the recovery time in the event of a system failure. If the checkpoint interval is small, then relatively few changes to the database are made between checkpoints and consequently, few changes need to be made during recovery. To speed up recoveries, create checkpoints frequently; note, however, that the server performance is reduced during the creation of a checkpoint. Furthermore, the speed of checkpoint creation depends on the amount of database cache used; the more database cache is used, the longer the checkpoint creation will take. See Appendix A, "Server-side configuration parameters," on page 117, for a description of the use of **CacheSize** parameter. You need to consider these issues when deciding the frequency of checkpoints.

For more details on checkpoints, read "Creating checkpoints" on page 37. You may also wish to read about transaction logging.

Reducing Bonsai Tree size by committing transactions

solidDB provides a consistent view of data within one transaction. If a user does not commit a transaction, solidDB keeps an image of the database as it existed at the moment the transaction was started — even if the transaction is a read-only transaction. This is implemented by the multiversioning solidDB Bonsai Tree, which stores the newest data in central memory. The new data is merged to the main storage tree as soon as currently active transactions no longer need to see the old versions of the rows.

When other connections perform many write operations, the server must use a large amount of memory to provide a consistent image of the database. If an open transaction remains uncommitted for a long duration of time, solidDB requires more memory; if the amount of memory available is insufficient, then solidDB performs excessive paging or swapping, which slows performance.

To determine whether slow performance is caused by excessive Bonsai Tree growth, you can monitor memory usage and Bonsai Tree size using operating system specific and solidDB specific tools.

Preventing excessive Bonsai Tree growth

To prevent excessive Bonsai Tree growth, make sure that every database connection commits every transaction. Even read-only transactions and transactions that contain only SELECT statements must be committed explicitly. (In autocommit mode, solidDB ODBC Driver version 3.50 and solidDB JDBC Driver version 2.0 perform an implicit commit after the last open cursor has been closed or dropped. In previous versions, the implicit commit is not available.)

Note that even in autocommit mode, SELECT statements are not automatically committed after the data is read. solidDB cannot immediately commit SELECTs since the rows need to be retrieved by the client application first. Even in autocommit mode, you must either explicitly commit work, or you must explicitly close the cursor for the SELECT statement. Otherwise, the SELECT transaction is left open until the connect timeout expires.

In order to ensure that every transaction is committed, you can:

- Determine what connections currently exist
- Determine when the connections have a committed transaction

- In the application code, ensure that every database operation gets committed
- Check for commit problems when using solidDB APIs

Each of these topics is described in the following sections.

Determining currently existing connections

The following solidDB commands and files allow you to determine the status of existing connections.

Table 23. Determinig command status

Command/File	Information
ADMIN COMMAND 'ul'	Obtain a list of existing connections.
ADMIN COMMAND 'sta'	Obtain the number of existing connections.
`solmsg.out`	Obtain the date and time when new connections are created.
ADMIN COMMAND 'trace on sql'	Obtain information when new connections are started. The results are written to the `soltrace.out` file.
ADMIN COMMAND 'report *filename*.txt'	Obtain a list of internal variables containing connection and status information.

Determining when connections have committed transactions

The following solidDB commands and files allow you to determine which connections have committed transactions.

Table 24. Determining which connections have committed transactions

Command/File	Information
ADMIN COMMAND 'trace'	Shows if a transaction gets committed at the server
ADMIN COMMAND 'report *filename*.txt'	Obtain a list of internal variables containing connection and status information. To find out connections that have not committed their transaction, look for the Readlevel for each connection. If the transaction at a particular connection is properly closed, the Readlevel should be zero (0) for that connection. To find those statements with active status, look under USER SEARCHES with column 'Act' having a value of 1. If the active status remains at the same Readlevel for a lengthy period of time, this is an indication that the statement has not closed or committed during this interval.

Providing commit statements in the application code

To make sure every database operation gets committed, be sure to either:

- Execute the statement COMMIT WORK.
- Call ODBC function SQLTransact or SQLEndTran.
- Call JDBC method commit.

Make sure these operations succeed by checking the return code or by properly catching the possible exception. Be aware how many database connections your application has, when and where they are created, and when the transactions at these connections are committed.

Troubleshooting COMMITs when using ODBC Driver Manager

When using ODBC Driver Manager and running in autocommit mode, most versions of ODBC Driver Manager regard calls to SQLTransact and SQLEndTran as redundant and never actually pass them to the driver.

This means that the application program only receives return code 'SUCCESS' from the ODBC Driver Manager, even though no transaction is committed in the database. This situation may go unnoticed. Besides, ODBC Driver Manager and SQL Editor, other utilities can also have an open transaction.

Make sure that you are aware of all database connections. Note that each FETCH after COMMIT (keeping the statement handle alive) also causes a new transaction to start.

Diagnosing poor performance

Different areas in solidDB can degrade its performance. To remedy performance problems, you need to determine the underlying cause. The following table lists common symptoms of poor performance, possible causes, and directs you to the section in this chapter for the remedy.

Table 25. Diagnosing poor performance

Symptoms	Diagnosis	Solution
Slow response time for a single query. Other concurrent access to the database is affected. Disk may be busy.	Inefficient usage of indexes in the query. Non-optimal decision from the Optimizer. External sorting is not defined and a large internal sorting is causing excessive swapping to disk.	If index definitions are missing, create new indices or modify existing ones to match the indexing requirements of the slow query. For more details, read the section in *IBM solidDB SQL Guide* titled "Using Indexes to Improve Query Performance". Run the EXPLAIN PLAN FOR statement for the slow query and verify whether the query optimizer is using the indices. For more details, read the description of the EXPLAIN PLAN FOR command in *IBM solidDB SQL Guide*. If the Optimizer is not choosing the optimal query execution plan, override the Optimizer decision by using optimizer hints. For more details, read "Using Optimizer Hints" in *IBM solidDB SQL Guide*. Make sure the external sorter is enabled by defining the Sorter.TmpDir configuration parameter. For more details, see “TmpDir_[1...N]” on page 53.
Slow response time is experienced for all queries. An increase in the number of concurrent users deteriorates the performance more than linearly. When all users are thrown out and then reconnected, performance still does not improve.	Insufficient cache size.	Increase the cache size. Allocate for cache at least 0.5MB per concurrent user or 2-5% of the database size. For more details, read “Dynamically changing database cache size” on page 90.

Table 25. Diagnosing poor performance (continued)

Symptoms	Diagnosis	Solution
Slow response time is experienced for all queries and write operations. When all users are thrown out and are connected, performance only improves temporarily. The disk is very busy.	The Bonsai Tree is too large to fit into the cache.	Make sure that there are no unintentionally long-running transactions. Verify that all transactions (also read-only transactions) are committed in a timely manner. For more details, read “Reducing Bonsai Tree size by committing transactions” on page 94.
Slow performance during batch write operation as the database size increases. There is an excessive amount of disk I/O.	The data is committed to the database in batches that are too small. Data is written to disk in an order that is not supported by the primary key of the table.	Make sure that the autocommit is switched off and the write operations are committed in batches of at least 100 rows per transaction. Modify the primary keys or batch write processes so that write operations occur in the primary key order. For more details, read chapter "Optimizing Batch Inserts and Updates" in *IBM solidDB SQL Guide*.
The server process footprint grows excessively and causes the operating system to swap. The disk is very busy. The ADMIN COMMAND 'report' output shows a long list of currently active statements.	SQL statements have not been closed and dropped after use.	Make sure that the statements that are no longer in use by the client application are closed and dropped in a timely manner.

6 Managing network connections

As a true client/server DBMS, solidDB provides simultaneous support for multiple network protocols and connection types. Both the database server and the client applications can be simultaneously connected to multiple sites using multiple different network protocols.

This chapter describes how to set up network connections for each of the supported platforms.

Note:

Some platforms and configurations may limit the number of concurrent users to a single solidDB server process even if the solidDB license accepts higher limits.

Communication between client and server

The database server and client transfer information between each other through the computer network using a communication protocol.

When a database server process is started, it will publish at least one network name that distinguishes it in the network. The server starts to *listen* to the network using the given network name. The network name consists of a communication protocol and a server name.

To establish a connection from a client to a server they both have to be able to use the same communication protocol. The client has to know the network name of the server and often also the location of the server in the network. The client process uses the network name to specify which server it will *connect* to.

This chapter will give you information on how to administer network names.

Managing network names

The network name of a server consists of a *communication protocol* and a *server name*. This combination identifies the server in the network. The network names are defined with the **Listen** parameter in the [Com] section of the configuration file. The `solid.ini` file should be located in a solidDB program's working directory or in the directory set by the SOLIDDIR environment variable.

A server may use an unlimited number of network names. Note that all components of network names are case insensitive.

Network names are managed in the following ways:

- Using the ADMIN COMMAND 'parameter com.listen' command in solidDB SQL Editor (solsql).
- Editing the server configuration file `solid.ini`.

An example of an entry in `solid.ini` is:

```
[Com]
Listen = tcpip 1313, nmpipe soliddb
```

The example contains two network names which are separated by a comma. The first one uses the protocol TCP/IP and the service port 1313; the other one uses the Named Pipes protocol with the name 'soliddb'. In the example, the 'tcpip' and 'nmpipe' are communication protocols, while '1313' and 'soliddb' are server names. The conventions for server names depend upon the protocol. A server name may be a name, such as "soliddb" or "chicago_office". A server name might be a service port number optionally preceded by a node name, such as "hobbes 1313" or "localhost 1313". In some protocols, the server name might simply be a service port number, such as "1313", if the client and server are running on the same computer.

If the **Listen** parameter is not set in the `solid.ini` file, the environment-dependent defaults are used.

Note:

1. When a database server process is started, it publishes the network names that it starts to listen to. This information is also written to a file named `solmsg.out` located in the same directory as the `solid.ini` file.
2. Network names must be unique within one host computer. For example, you cannot have two database servers running, both listening to the same TCP/IP port in one host, but it is possible that the same port number is in use in different hosts. Exceptions to this is the NetBIOS protocol, which requires that the used server names are unique throughout the whole network.

Viewing supported protocols for the server

All protocols are not supported in all environments and operating systems.

To view supported protocols for your server, enter the following command in solidDB SQL Editor (solsql):

ADMIN COMMAND 'protocols'

A list of all available communication protocols is displayed. The command provides the following kind of result set, which contains one row for each supported communication protocol:

```
admin command 'protocols';
    RC TEXT
    -- ----
    0 NetBIOS    nb
    0 NmPipe     np
    0 TCP/IP     tc
3 rows fetched.
```

Viewing network names for the server

You can view the network names for the server in the following ways:

- View the **Listen** parameter in the [Com] section in the `solid.ini` file.
- Enter the following command in solidDB SQL Editor (solsql):

  ADMIN COMMAND 'parameter com.listen'

  A list of all network names for the server is displayed.

Adding and modifying a network name for the server

Following are ways you can add and edit network names for a server, which consists of a *communication protocol* and a *server name*; for example, nmpipe soliddb.

- To add network names for the server, enter the following command in solidDB SQL Editor (solsql):

ADMIN COMMAND 'parameter com.listen= *network_name*'

The command returns the new value as the resultset. If the network name entered is invalid, the ADMIN COMMAND statement returns an error. Otherwise the new name is enacted immediately. The changes are written to `solid.ini` at the next checkpoint.

- In `solid.ini`, locate the working directory of your solidDB process and add a new network name or edit an existing one as a part of the **Listen** parameter entry in the [Com] section.

  Use a comma (,) to separate network names. For example:

  ```
  [Com]
  Listen = tcpip 1313, nmpipe soliddb
  ```

  Be sure to save the changes and to restart the solidDB process to activate the changes.

Removing network name from the server

You can remove network names for a server, which consists of a *communication protocol* and a *server name*, for example, nmpipe soliddb, as follows:

- To make the change by updating the `solid.ini` configuration file, locate the working directory of your solidDB process and remove the network name in the **Listen** parameter entry in the [Com] section.

  Be sure to save the changes and to restart the solidDB process to activate the changes.

When you start the server, if you want to temporarily disable one of the network names listed in the `solid.ini` file, you can disable the network name by using option -d after the protocol name in the network name when you start the server. For example:

```
solid tcp -d hobbes 1313
```

This prevents the server from using this network name. This does not change the contents of the `solid.ini` file, so this will have no effect on the server name(s) the next time that the server starts up.

Factory value for a network name

If no network name is specified in the .ini file, the server uses a factory preset that is "tcpip 1964". In other words, the server will listen to the TCP/IP port 1964, if no .ini file is used.

Connect strings for clients

A networks name used by a client is a logical data source name or a data source connect string. A data source connect string consists of a *communication protocol*, a possible set of *special options*, an optional *host computer name* and a *server* name. By this combination, the client specifies the server it will establish a connection to. The communication protocol and the server name must match the ones that the server is using in its network listening name. In addition, most protocols need a specified host computer name if the client and server are running on different machines. All components of the client's network name are case insensitive.

The same format of a connect string for clients applies to both the connect configuration parameters in the `solid.ini` file and network names used in ODBC applications.

The format of a connect string is the following:

```
protocol_name [options] [server_name] [port_number]
```

where options may be any number of:

Table 26. Connect string format

Option	Meaning
-z	Data compression is enabled for this connection
-c *milliseconds*	Login timeout is specified (the default is operating-system-specific). A login request fails after the specified time has elapsed. Note: for the tcp protocol only.
-r *milliseconds*	Connection (or read) timeout is specified (the default is 60 s). A network request fails when no response is received during the time specified. The value 0 sets the timeout to infinite. Note: applies for the tcp protocol only.

Examples:

```
tcp localhost 1315
tcp 1315
tcp -z -c1000 1315
nmpipe host22 SOLIDDB
```

Mapping logical data source names to connect strings

solidDB Clients support Logical Data Source Names. These names can be used for giving a database a descriptive name. This name can be mapped to a data source in three ways:

1. Using the parameter settings in the application's `solid.ini` file.
2. Using the Microsoft Windows operating system's registry settings.
3. Using settings in a `solid.ini` file located in the Windows directory.

This feature is available on all supported platforms. However, on non-Windows platforms, only the first method is available.

A solidDB Client attempts to open the file `solid.ini` first from the directory set by the SOLIDDIR environment variable. If the file is not found from the path specified by this variable or if the variable is not set, an attempt is made to open the file from the current working directory.

To define a Logical Data Source Name using the `solid.ini` file, you need to create a `solid.ini` file containing the section [Data Sources]. In that section you need to enter the 'logical name' and 'network name' pairs that you want to define. The syntax of the parameters is the following:

```
[Data Sources]
logical_name = connect_string, Description
```

In the description field, you may enter comments on the purpose of this logical name.

For example, assume you want to define a logical name for the application My_application and the database that you want to connect is located in a UNIX server using TCP/IP. Then you should include the following lines in the `solid.ini` file, which you need to place in the working directory of your application:

```
[Data Sources]
My_application = tcpip irix 1313, Sample data source
```

When your application now calls the Data Source 'My_application', the solidDB Client maps this to a call to 'tcpip irix 1313'.

On Windows platforms, the registry is typically used to map Data Sources. To setup the registry with a GUI interface, use the Windows Administrative Control Panel "Data Sources (ODBC)".

Default connect string

When no data source is specified for the connection, the default connect string will be used. The client's default connect string may be defined in the client's configuration file `solid.ini` in the [Com] section with the Connect parameter. The client's `solid.ini` file should be located in the application program's working directory or in the directory set by the SOLIDDIR environment variable. The value of the Connect parameter is read by all solidDB tool programs and client libraries when no data source is specified for the connection. The client libraries do not need this value if a valid connect string is supplied at run time, or when a standard ODBC driver manager is used.

The following connect line in the `solid.ini` of the application workstation will connect an application (client) using the TCP/IP protocol to a solidDB server running on a host computer named 'spiff' and listening with the name (port number in this case) '1313'.

```
[Com]
Connect = tcpip spiff 1313
```

If the Connect parameter is not found in the `solid.ini` configuration file, then the client uses the environment-dependent default instead. The defaults for the Listen and Connect parameters are selected so that the application (client) will always connect to a local solidDB server listening with a default network name. So local communication (inside one machine) does not necessarily need a configuration file for establishing a connection.

Communication protocols

A client process and solidDB communicate with each other by using computer networks and network protocols. Supported communication protocols depend on the type of computer and network you are using.

The following paragraphs describe the supported communication protocols and common environments that may be used and also show the required forms of network names for the various protocols.

Note:

Depending on your network protocol, there may be relevant communication parameters associated with the protocol. Be sure to use ADMIN COMMAND 'parameter' in the solidDB Query window to find the communication parameters in use. Then you can use ADMIN COMMAND 'describe parameter' to view details on the specific communication parameter. See 3, “Configuring solidDB,” on page 47 for details on these commands.

Shared Memory

Note: Shared Memory is has been deprecated as of release 6.3 Fix Pack 1.

Usually the fastest way two processes can exchange information is to use Shared Memory. This can be used only when solidDB and application processes are both running in the same computer. The Shared Memory protocol uses a shared memory location for moving data from one process to another.

To use the Shared Memory protocol in solidDB, select ShMem from the list of protocols in solidDB and enter server name. The server name has to be unique only in this computer.

Table 27. Shared Memory protocol in the `solid.ini` file

Where	Syntax example
Server	`Listen = shmem servername`
Client	`Connect = shmem servername`

Note:

Server names must be character strings less than 128 characters long.

TCP/IP

When starting a server using the TCP/IP protocol, you must reserve a port number for it. You will find reserved port numbers in the `/etc/services` file of your system. Select a free number greater than 1024 since smaller numbers are usually reserved for the operating system.

To use the TCP/IP protocol, select TCP/IP in the list of protocols in solidDB and enter a non-reserved port number.

Table 28. TCP/IP protocol in the `solid.ini` file

Where	Syntax example
Server	`Listen = tcpip server_port_number`
Client	`Connect = tcpip [host_computer_name] server_port_number`

For example

```
Listen = tcp 1315
Connect = tcpip accounting_dept_server 1315
```

Note:

1. If the server is running in the same computer with the client program, the host computer name need not be specified. The client computer must have the used host name listed in its `/etc/hosts` file or it must be recognized by the DNS (Domain Name Server). You can also give the host computer's TCP/IP address in dotted decimal format (for example, 194.53.94.97) instead of its host name.

2. On Windows and UNIX, the TCP/IP protocol is usually included in the operating system. On other environments (like VAX/VMS) the TCP/IP software needs to be installed on the system. For a list of supported TCP/IP software, contact IBM Corporation at: http://www.ibm.com/software/data/soliddb.
3. The local loopback interface address, 127.0.0.1, is the default address when a client attempts to open a TCP/IP connection without specifying a hostname.
4. Using option -i *ip_address* or -i *host_name* , the solidDB listens only to the specified IP address or host name. This is useful in multi-homed systems that support many TCP/IP interfaces (or have multiple ip-addresses). For example, a server with the following setting in `solid.ini` accepts connection requests only from inside the same machine, either referred by IP address 127.0.0.1 or with the name 'localhost', if the DNS is correctly configured:

   ```
   [com]
   Listen = tcp -i127.0.0.1 1313
   ```

   Note that DNS entries can be used instead of IP addresses, for example:

   ```
   [com]
   Listen = tcp -ilocalhost 1313
   ```
5. Using option -i127.0.0.1, which starts the server to listen only to a local loopback connection, allows TCP/IP listening with a desktop license. To enable TCP/IP usage with desktop licenses, all entries in `solid.ini` have to be edited to include -i. Note that default listening of port 1313 (without `solid.ini`) works automatically.

UNIX Pipes

The UNIX domain sockets (UNIX Pipes) are typically used when communicating between two processes running in the same UNIX machine. UNIX Pipes usually have a very good throughput. They are also more secure than TCP/IP, since Pipes can only be accessed from applications that run on the computer where the server executes.

When starting a server using UNIX Pipes, you must reserve a unique listening name (inside that machine) for the server, for instance, 'soliddb'. Because UNIX Pipes handle the UNIX domain sockets as standard file system entries, there is always a corresponding file created for every listened pipe. In solidDB's case, the entries are created under the path `/tmp` Our example listening name 'soliddb' creates the directory `/tmp/solunp_SOLIDDB` and shared files in that directory. The `/tmp/solunp_` is a constant prefix for all created objects while the latter part ('SOLIDDB' in this case) is the listening name in upper case format.

Table 29. UNIX Pipes protocol in the `solid.ini` file

Where	**Syntax example**
Server	`Listen = upipe` *`server_name`*
Client	`Connect = upipe` *`server_name`*

Note:

1. Server and client processes must run in the same machine in order to use UNIX Pipes for communication.
2. The server process must have "write" permission to the directory `/tmp`.

3. The client that is accessing UNIX Pipes must have "execute" permission on the directory `/tmp`.
4. The directory `/tmp` must exist.
5. UNIX Pipes cannot be used in Caldera/SCO UNIX.

Named Pipes

Named Pipes is a protocol commonly used in the Microsoft Windows operating systems.

Table 30. Named Pipes protocol in the `solid.ini` file

Where	Syntax example
Server	`Listen = nmpipe` *`server_name`*
Client	`Connect = nmpipe [` *`host_computer_name`* `]` *`server_name`*

Note:

1. The server names must be character strings at most 50 characters long.
2. If the server is running in the same computer with the application program, the host computer name should not be specified.
3. In order to connect to the solidDB for Windows through Named Pipes, the user must have at least the same rights as the user who started the server. For example if an administrator starts the server, then only users with administrator's rights are able to connect to the server through Named Pipes. Similarly, if a user with normal user's rights starts the server, then all users with equal or greater rights are able to connect the server through Named Pipes. If a user doesn't have proper rights, solidDB Communication Error 21306 message will be given.
4. It is not recommended to use the Named Pipes communication from solidDB Remote Control. The asynchronous nature of solidDB Remote Control communication may cause problems with Named Pipes.

Note that you may use either "nmpipe" or "nmp" to specify the named pipes protocol.

NetBIOS

The NetBIOS protocol is commonly used in the Microsoft Windows operating systems.

To use NetBIOS protocol, select NetBIOS in the list of available protocols in solidDB and enter a non-reserved server name.

Table 31. NetBIOS protocol in the `solid.ini` file

Where	Syntax example
Server	`Listen = netbios [a`*`LANA_NUMBER`*`]` *`server_name`*
Client	`Connect = netbios [a`*`LANA_NUMBER`*`]` *`server_name`*

Note:

1. The server name must be a character string at most 16 characters long. It may not begin with an asterisk (*).
2. In the above format, the optional -aLANA_NUMBER parameter is used to override the default value of the LANA number.
3. In Windows, the available LANA numbers can be checked using the Network Setup found in the Control Panel. The default value 0 may not be generally very good. You should choose the one(s) where the protocol stack matches the other computers you are using. The LANA number (Network Route: Nbf → Elnk3 → Elnk31) that uses NetBEUI as a transport usually functions quite smoothly when used for solidDB communication.
4. The server names have to be unique in the whole network. Establishing a connection or starting the listener using the NetBIOS protocol may be somewhat slow because of the checks needed for uniqueness.
5. solidDB products use all available LANA numbers by default. This makes it unnecessary to specify explicitly which LANA number the application or solidDB should use. For backward compatibility, the -aLANA_NUMBER parameter remains available.

Summary of protocols

The following tables summarize the possible operating systems and required forms for network names for the various communication protocols.

Table 32. solidDB protocols and network names

Protocol	**Server OS**	**Network name in `solid.ini` file**
Shared Memory - deprecated as of release 6.3 Fix Pack 1	Windows	Listen = shmem *server*
NetBIOS	Windows	Listen = netbios *server*
Named Pipes	Windows	Listen = nmpipe *server*
TCP/IP	Windows, UNIX, VxWorks	Listen = tcpip *port*
UNIX Pipes	UNIX	Listen = upipe *server*

Table 33. Application protocols and network names

Protocol	**Server OS**	**Network name in `solid.ini` file**
Shared Memory - deprecated as of release 6.3 Fix Pack 1	Windows	Connect = shmem *server*
NetBIOS	Window	Connect = netbios *server*
Named Pipes	Windows	Connect = nmpipe [host] *server*
TCP/IP	Windows, UNIX, VxWorks	Connect = tcpip [host] *port*
UNIX Pipes	UNIX	Connect = upipe *server*

1) Requires Digital PATHWORKS 32 for Microsoft Windows.

Logical Data Source Names

solidDB Clients support Logical Data Source Names. These names can be used for giving a database a descriptive name. This name can be mapped to a network name in three ways:

1. Using the parameter settings in the application's `solid.ini` file.
2. Using the Microsoft Windows operating system's registry settings.
3. Using settings in a `solid.ini` file located in the Windows directory.

This feature is available on all supported platforms. However, on non-Windows platforms, only the first method is available.

A solidDB Client attempts to open the file `solid.ini` first from the directory set by the SOLIDDIR environment variable. If the file is not found from the path specified by this variable or if the variable is not set, an attempt is made to open the file from the current working directory.

To define a Logical Data Source Name using the `solid.ini` file, you need to create a `solid.ini` file containing the section [Data Sources]. In that section you need to enter the *logical name* and *network name* pairs that you want to define. The syntax of the parameters is the following:

```
[Data Sources]
logical_name = network_name, Description
```

In the description field, you may enter comments on the purpose of this logical name.

For example, assume you want to define a logical name for the application My_application and the database that you want to connect is located in a UNIX server using TCP/IP. Then you should include the following lines in the `solid.ini` file, which you need to place in the working directory of your application:

```
[Data Sources]
My_application = tcpip irix 1313, Sample data source
```

When your application now calls the Data Source My_application, the solidDB Client maps this to a call to 'tcpip irix 1313'.

On Windows platforms, the registry can be used to map Data Sources. These follow the standards of mapping ODBC Data Sources on a system.

In Windows, a Data Source may be defined in the Windows Registry. The entry is searched from the path `software\odbc\odbc.ini`

1. first under the root HKEY_CURRENT_USER and if not found,
2. under the root HKEY_LOCAL_MACHINE.

The order of resolving a Data Source name in Microsoft Windows systems is the following:

1. Look for the Data Source Name from the `solid.ini` file in the current working directory, under the section [Data Source]
2. Look for the Data Source Name from the following registry path HKEY_CURRENT_USER\software\odbc\odbc.ini\DSN
3. Look for the Data Source Name from the following registry path HKEY_LOCAL_MACHINE\software\odbc\odbc.ini\DSN

If an application uses normal ODBC Data Sources, the network name is mapped normally using the methods that are provided in the ODBC Driver Manager.

7 Diagnostics and troubleshooting

This chapter provides information on the following solidDB diagnostic tools:

- Network trace facility used to trace the server communication
- Ping facility used to trace client communication

You can use these facilities to observe performance, troubleshoot problems, and produce high quality problem reports. These reports let you pinpoint the source of your problems by isolating them under product categories (such as solidDB ODBC API, solidDB ODBC Driver, solidDB JDBC Driver, etc.).

You may also want to read "Performance counters (perfmon)" on page 19, which discusses various monitoring techniques including the perfmon command.

Tracing communication between client and server

solidDB provides the following tools for observing the communication between an application or an external application (if using linked library access) and a database server:

- the Network Trace facility
- the Ping facility

You can use these tools to analyze the functionality of the networking between an application and solidDB. The network trace facility should be used when you want to know why a connection is not established to solidDB. The ping facility is used to determine how fast packets are transferred between an application and a database server.

The network trace facility

Network tracing can be done on the solidDB computer, on the application computer or on both computers concurrently. The trace information is written to the default trace file or file specified in the **TraceFile** parameter.

The default name of the output file is `soltrace.out`. This file will be written to the current working directory of the server or client depending on which end the tracing is started.

The file contains information about:

- loaded DLLs
- network addresses
- possible errors

The Network Trace facility is turned on by editing the configuration file:

```
[Com]
Trace ={Yes|No}
; default No
TraceFile = file_name
; default soltrace.out
```

or by using the environment variables SOLTRACE and SOLTRACEFILE to override the definitions in the configuration file. Setting of SOLTRACE and SOLTRACEFILE environment variables have the same effect as the parameters Trace and TraceFile in the configuration file.

Note: Defining the **TraceFile** configuration parameter or the SOLTRACEFILE environment variable automatically turns on the Network trace facility.

A third way to turn on the Network trace facility is to use the option `-t` and/or `-ofilename` as a part of the network name. The option `-t` turns on the Network trace facility. The option `-o` turns on the facility and defines the name of the trace output file.

Defining parameter Trace in the client-side configuration file

```
[Com]
Connect = nmp SOLIDDB
Listen = nmp SOLIDDB
Trace = Yes
```

Defining environment variables

```
set SOLTRACE = Yes
```

or

```
set SOLTRACEFILE = trace.out
```

Using network name options

```
[Com]
Connect = nmp -t soliddb
Listen = nmp -t soliddb
```

or

```
[Com]
Connect = nmp -oclient.out soliddb
Listen = nmp -oserver.out soliddb
```

Network trace facility output

Following is an excerpt from a trace file:

```
Scanning listening keyword Listen from section Com.
No listening information found from section Com.
Generating default listening info.

Parsing address 'TCP/IP 1964'.
Address information:
        fullname : 'TCP/IP 1964'
        lisname  : '1964'
        protocol : 'tcp' (TCP/IP)
        enabled  : Yes
        ping     : 0
        trace    : No

Reading communication configuration from file D:\solid\solid.ini.

Parsing address 'TCP/IP 1964'.
Address information:
        fullname : 'TCP/IP 1964'
        lisname  : '1964'
        protocol : 'tcp' (TCP/IP)
        enabled  : Yes
        ping     : 0
```

```
        trace    : No

Initialising protocol 'tcp' (TCP/IP).
Searching DLL 'DTCW3237'.
DLL s:\soldll\DTCW3237.DLL loaded.
SOLID version 03.70.0026, DLL interface version 4.
Build information Tue Oct 25 00:18:07 2002.
Initialization of protocol 'tcp' succeeded.

Protocol TCP/IP using configuration :
MaxPhysMsgLen: 8192
  ReadBufSize: 2048
 WriteBufSize: 2048
 SelectThread: Yes
        Trace: Yes
   MinWritePoolBuffers: 4
   MaxWritePoolBuffers: -1
    WritePoolIncrement: 1
     SyncRead: No
    SyncWrite: No

26.07 15:12:21 Initializing server. Listen info 'TCP/IP 1964'.
Starting the listening of 'TCP/IP 1964'.
```

The Ping facility

The Ping facility can be used to test the performance and functionality of the networking. The Ping facility is built into all solidDB client applications and is turned on with the network name option -p *level* .

The output file will be written to the current working directory of the computer where the parameter is given. The default name of the output file is `soltrace.out`.

Clients can always use the Ping facility at level 1. Levels 2, 3, 4 or 5 may only be used if the server is set to use the Ping facility at least at the same level.

The Ping facility levels are:

Table 34. Ping facility levels

Setting	Function	Description
0	no operation	do nothing, default
1	check that server is alive	exchange one 100 byte message
2	basic functional test	exchange messages of sizes 0.1K, 1K, 2K..30K, increment 1K
3	basic speed test	exchange 100 messages of sizes 0.1K, 1K, 8K and display each sub-result and total time
4	heavy speed test	exchange 100 messages of sizes 0.1K, 1K, 2K, 4K, 8K, 16K and display each sub-result and total time
5	heavy functional test	exchange messages of sizes 1..30K, increment 1 byte

Note:

If a solidDB client does not have an existing server connection, you can use the SQLConnect() function with the connect string -p1 option (ping test, level 1) to check if solidDB is listening in a certain address. Without logging into solidDB, SQLConnect() can then check the network layer and ensure solidDB is listening. When used in this manner, SQLConnect() generates error code 21507, which means the server is alive.

Running Ping facility at level 1

The client turns on the Ping facility by using the following network name:

```
nmp -p1 -oping.out SOLIDDB
```

This runs the Ping facility at the level 1 into a file named `soltrace.out`. This test checks if the server is alive and exchanges one 100 byte message to the server.

After the Ping facility has been run, the client exits with the following message:

```
SOLID Communication return code xxx: Ping test successful/failed,
results are in file FFF.XX
```

How the listen parameter restricts the use of Ping facility

If the server is using the following listen parameter, applications can run the Ping facility at levels 1, 2, and 3, but not 4 and 5.

```
[Com]
Listen = nmp -p3 SOLID
```

Note:

Ping clients running at level greater than 3 may cause heavy network traffic and may slow down any application that is using the network, including any ordinary SQL clients connected to the same solidDB.

Problem reporting

solidDB offers sophisticated diagnostic tools and methods for producing high quality problem reports with very limited effort. Use the diagnostic tools to capture all the relevant information about the problem.

All problem reports should contain the following files and information:

- `solid.ini`
- license number
- `solmsg.out`
- `solerror.out`
- `soltrace.out`
- `ssdebug.out`
- problem description
- steps to reproduce the problem
- all error messages and codes
- contact information, preferably email address of the contact person

Problem categories

Most problems can be divided into the following categories:

- solidDB ODBC API
- solidDB ODBC or JDBC Driver
- Communication problems between the application or an external application (if using linked library access) and solidDB.
- Problems with disk block integrity

The following pages include detailed instructions to produce a proper problem report for each problem type. Please follow the guidelines carefully.

solidDB ODBC API problems

If the problem concerns the performance of a specific solidDB ODBC API or SQL statement, you should run SQL info facility at level 4 and include the generated `soltrace.out` file into your problem report. This file contains the following information:

- create table statements
- create view statements
- create index statements
- SQL statement(s)

solidDB ODBC Driver problems

If the problem concerns the performance of solidDB ODBC Driver, please include the following information:

- solidDB ODBC Driver name, version, and size
- ODBC Driver Manager version and size

If the problem concerns the cooperation of solidDB and any third party standard software package, please include the following information:

- Full name of the software
- Version and language
- Manufacturer
- Error messages from the third party software package

Use ODBC trace option to get a log of the ODBC statements and include it in your problem report.

solidDB JDBC Driver problems

If the problem is related to the solidDB JDBC Driver, please include the following information in your problem report:

- Exact version of JDK or JRE used
- Name, size, and date of the SOLIDDriver class package
- Contents of DriverManager.setLogStream(someOutputStream) output, if available
- Call stack (that is, Exception.printStackTract() output) of the application, if an exception has occurred in the application

Communication between a client and server

If the problem concerns the performance of the communication between a client and server use the Network trace facility and include the generated trace files into your problem report. Please include the following information:

- solidDB communication DLLs used: version and size
- Other communication DLLs used: version and size
- Description of the network configuration

Database disk block integrity

If the problem concerns the database disk block integrity, check the integrity by starting solidDB database with the -x testblocks parameter. This option will check the disk block integrity and produce a report in the `ssdebug.out` file.

Appendix A. Server-side configuration parameters

By managing the configuration parameters of your solidDB, you can modify the environment, performance, and operation of the server. The configuration parameters are stored in the `solid.ini` configuration file and are read when the server starts.

Generally, the factory value settings offer the best performance and operability, but in some special cases modifying a parameter will improve performance. You can change the parameters in the following ways:

- Manually editing the configuration file `solid.ini`. Since the file is only read when the server is started, changes to a parameter value in the `solid.ini` file do not take effect until the next time that the server is started.
- Entering the command

  ```
  ADMIN COMMAND 'parameter name=value'
  ```

Note: Parameter support may vary between platforms.

The first part of this appendix focuses on the `solid.ini` file, and describes the proper format for parameter values in that file.

The second part of this appendix describes how to use an ADMIN COMMAND to change the value of a parameter dynamically.

The remainder of this appendix describes the parameters themselves, including the valid range of values and the factory values.

Note: Parameters for some components, such as HotStandby, may be described in the manual for that component rather than in this guide.

Setting parameters through the `solid.ini` configuration file

When the solidDB is started, it attempts to open the configuration file `solid.ini`. If the file does not exist, solidDB will use the factory values for the parameters. If the file exists, but a value for a particular parameter is not set in the `solid.ini` file, solidDB will use a factory value for that parameter. The factory values depend on the operating system you are using.

By default, the server looks for the `solid.ini` file in the current working directory, which is normally the directory from which you started the server. If you would like to specify a different directory to be used as the current working directory, then use the -c command line option. (For more details about command line options, see Appendix C, “solidDB command line options,” on page 175.) If you want to specify a different directory for the `solid.ini` file, you can set the SOLIDDIR environment variable to specify the location of the `solid.ini` file. When searching for the file, the solidDB uses the following precedence (from high to low):

- location specified by the SOLIDDIR environment variable (if this environment variable is set)
- current working directory

Rules for formatting the solid.ini file

The configuration file solid.ini is an ASCII file with line breaks.

The solid.ini configuration file is divided into sections. Each section contains a group of one or more loosely-related parameters. Each section has a name, and that name is delimited with square brackets, e.g.

```
[SQL]
```

Within each section are the parameters. Parameters are specified in the following format:

```
param_name=param_value
```

for example:

```
Listen=tcp 127.123.45.156 1313
DurabilityLevel=2
```

Blank spaces around the equals sign are allowed but not required. The following are equivalent:

```
DurabilityLevel=2
DurabilityLevel = 2
```

If you omit the parameter value, then the server will use the factory value. For example:

```
; Use the factory value
DurabilityLevel=
```

If you omit the parameter value and the equals sign, you get an error message.

Every parameter must be under a section header. If you put a parameter before any section header, you get an error message indicating that there is an unrecognized entry in the section named "<no section>".

Section names can be repeated. For example:

```
[Index] BlockSize=2048
[Com]
...
[Index]
CacheSize=8m
```

However, repeating sections names makes it more difficult for users to keep the file up-to-date and consistent, so we do not recommend doing this.

Parameter names can also be repeated (you won't get a warning message), but this is very strongly discouraged. The last occurrence of the parameter in the file takes the precedence.

The solid.ini file can contain comments, which must begin with a semicolon.

```
; This is a valid comment.
```

You can also put a comment on the same line as a parameter.

```
DurabilityLevel=2 ; This is also a valid comment.
```

Below is a simple example of part of a solid.ini file that contains a section heading, a parameter, and a comment:

```
[Logging]
; Use "relaxed logging", which improves performance but may
; risk losing the last few transactions during a failure.
DurabilityLevel=1

[Com]
...
```

There are a few cases where two or more sections have parameters with the same name. Therefore, you must be careful to place each parameter in the correct section.

Most sections and parameters are optional. You do not need to specify a value for every parameter in every section, and in fact you can omit entire sections. If you omit a parameter(s), the server will use the factory value. Later in this appendix, we list each section, each parameter name, the factory value for that parameter, and a description of the purpose and valid range of values for that parameter.

The server checks each entry in the `solid.ini` file. If the entry is not a comment, the server checks that the combination of section name and parameter name is valid. If you have invalid entries in the file, the server will display an error message in the `solmsg.out` file; if the server is running as a foreground process, the message will also be displayed on the console. The message will be similar to one of the following:

1. `Warning: Unrecognized entry in inifile: '<section>.<parameter>'.`

   You will see this message if you have entries that fit the proper form, but which do not have the pre-defined section names and parameter names. For example, you would get this message if you had a `solid.ini` file like the following:

   ```
   ; This has a valid section name, but an invalid parameter name.
   [Logging]
   NoSuchParam=NoSuchValue

   This has an invalid section name.
   [NoSuchSectionName]
   ```

   The message for the first of these errors would be similar to:

   ```
   Warning: Unrecognized entry 'Logging.NoSuchParam' in inifile.
   ```

2. `Warning: Illegal entry in inifile: <whole illegal line>`

   The server will display this message if a line could not be recognized as a section header, parameter name, comment, or blank line. You may see this message if you have entries that are not in the proper form. For example, you will see this message if your `solid.ini` file contains something like the following:

   ```
   ; This text was intended to be a comment
   but we forgot to precede part of it with a semicolon.
   ```

3. `Warning: 1 unrecognized or illegal entry in '<inifilename>'`

   or

   `Warning: <number> unrecognized or illegal entries in '<inifilename>'.`

   After the server has finished processing the `solid.ini` file, it will list the total number of errors detected.

4. `Warning: Unregistered parameter <section>.<parameter> is used.`

   If this error occurs, it is a sign of a possible problem inside the server itself. If you see this error, please report it to IBM Corporation.

Note that the server does not necessarily display an error message if you use an invalid value for a parameter. The server may simply use the factory value without issuing an error message.

The `solid.ini` parameter file is checked only when the server starts. If you edit it after the server starts, the server will not see the changes until the next time that the server starts.

CAUTION:
If you make changes to the `solid.ini` file AND you make changes to parameters in the server by using an ADMIN COMMAND, the behavior is unpredictable. While the server is running, you can safely change the `solid.ini` file OR make changes to server values using the ADMIN COMMAND, but you should not do both during the same "run" of the server.

A summary of the rules is below:

- Section name is in the format
  `[section-name]`
- The same section name may be used several times (however, this is not recommended).
- Each parameter is set in a separate line.
- Entries in the files may be preceded with blanks.
- If the first non-blank character is the comment character, then the whole line is ignored (that is, it is treated as a comment line).
- The comment character is the semicolon (;).
- Comments may follow other entries that are in the same line.
- Lines that have no characters, or that have only blank characters, are ignored.

Format of configuration parameter names and values

The rules for configuration parameter names and values are the same regardless of whether the parameters are set through the `solid.ini` file or an ADMIN COMMAND:

- The section and parameter names are not case-sensitive.
- The string values are not case-sensitive.
- In most cases, units are not case-sensitive. For example, to specify that the units are in megabytes, you may use any of the following: m, M, MB, mb, Mb, or mB. Some units (e.g. time units 's' (seconds) and 'ms' (milliseconds)) are case sensitive and such cases are documented.
- The syntax for general parameter value setting is:
  `param_name [space characters] = [space characters] value_literal`
  The syntax for the value is
  `value_literal [space characters] unit_of_measure`
  where
  param_name is the parameter name. When this is used in an ADMIN COMMAND, the name should be the full parameter name, including the section name, for example, **Logging.DurabilityLevel**. When this is used in the `solid.ini` file, it should NOT include the section name, since the parameter should already be listed under the appropriate section header.
  value_literal is the value to be assigned to the parameter. This is usually a literal, such as the number 12, or the string "tcp MyServer2 1315". If you give no value, the parameter will be set to its startup value. If you assign a parameter value

with an asterisk (*), the parameter will be set to its factory value. Note that string literals should normally be in double quotes if they are used in an ADMIN COMMAND.

unit_of_measure is the unit of measure, for example *MB* for megabytes or *ms* for milliseconds.

[space characters] represents places where spaces are allowed but not required. Spaces around the equals sign are optional. Spaces between the value and the unit of measure are optional.

For example, allowed forms include:

```
CacheSize=32M
cachesize=32m
CacheSize = 32 m
etc.
```

Changing parameters through ADMIN COMMAND

Most parameters can be changed with an ADMIN COMMAND:

```
ADMIN COMMAND 'parameter param_name = value [temporary]';
```

The *param_name* and *value* generally follow the rules specified in "Format of configuration parameter names and values" on page 120.

Note: If no value is specified, this sets the parameter with a factory (or unset) value. Furthermore, if you assign a parameter value with an asterisk (*), the parameter will be set to its factory value.

Note that the param_name in an ADMIN COMMAND (unlike in the `solid.ini` file) must include the section name and the parameter name, separated by a period character. For example, to set the value of the **DurabilityLevel** parameter, which is part of the [Logging] section, issue a command like:

```
ADMIN COMMAND 'parameter Logging.DurabilityLevel=1';
```

When the value of a parameter is changed with an ADMIN command, the change may or may not apply immediately, and may or may not apply the next time that the server is started. If a parameter value is written to the `solid.ini` file, then it will take effect the next time that the server starts. If the temporary option is used, then the value will affect the server's current behavior, but will not affect the server when it restarts. In some cases, changing a parameter may take effect immediately AND be written to the `solid.ini` file so that it also applies the next time that the server starts. See the explanations of Access Mode below.

Access Mode

The tables later in this appendix list the "Access Mode" for each parameter. The Access Mode indicates whether the parameter can be changed dynamically (via an ADMIN COMMAND), and when the change takes effect. The possible Access Modes are:

- RO (read-only): the value cannot be changed; the current value is always identical to the startup value.
- RW: can be changed via an ADMIN COMMAND, and the change takes effect immediately.
- RW/Startup: can be changed via an ADMIN COMMAND, and the change takes effect the next time that the server starts.

- RW/Create: can be changed via an ADMIN COMMAND, and the change applies when a new database is created.

Saving parameter changes

Unless the option temporary is used, all the changes made to the parameters will be saved in the `solid.ini` file at the next checkpoint. The saving may be also expedited with the command:

```
ADMIN COMMAND
'save parameters [file_name]';
```

By default, the command rewrites the default `solid.ini` file. By using the *file_name* option, the output can be directed to a different location.

Descriptions of configuration parameters

There is one table below for each section of the `solid.ini` file. The sections (and tables) are:

- Accelerator
- Cluster
- Com
- General
- HotStandby (discussed in the *IBM solidDB High Availability User Guide*)
- IndexFile
- Logging
- LogReader
- MME
- Sorter
- SQL
- Srv
- Synchronizer

Most parameters in most sections apply to all solidDB components. The sections that do not apply to all components are listed below:

- The MME section applies only to the solidDB diskless edition.
- The Synchronizer section applies only to solidDB advanced replication capability, which is available in the solidDB in-memory database.
- The HotStandby section only applies to the HotStandby component.

The descriptions of a few individual parameters specify that those parameters (or some specific settings of those parameters) apply only to a particular component. Each of these exceptions is documented in the description of the parameter itself.

Accelerator section

Table 35. Accelerator parameters

[Accelerator]	Description	Factory Value
ImplicitStart	If set to yes, this parameter starts solidDB automatically as soon as the ODBC API function SQLConnect is called in a user application. If set to no, solidDB must be explicitly started with a call to the Control API function SSCStartServer.	yes

Cluster section

Table 36. Cluster parameters

[Cluster]	Description	Factory Value	Access Mode
ReadMostlyLoadPercentAtPrimary	Percentage of read load directed to the Primary	50	RW/Startup

Communication section

Table 37. Communication parameters

[Com]	Description	Factory Value	Access Mode
Listen	Defines the network name for a server. When a solidDB database server process is started, it will publish at least one network name that distinguishes it in the network. The server can then start to listen to the network using the given network name. The network name consists of a communication protocol and a server name. For more details, read 6, "Managing network connections," on page 99.	tcp 1964	RW
MaxPhysMsgLen	Defines the maximum length of a single physical network message in bytes; longer network messages will be split into smaller messages of this size.	OS dependent	RW/Startup

Table 37. Communication parameters (continued)

[Com]	Description	Factory Value	Access Mode
RConnectLifetime	A time period in seconds for how long the idle connections are kept open in the pool. Whenever the connection is used, the timer starts from zero. Valid values range from 0-3600 This parameter is associated with server-maintained remote connections used to execute Remote Stored Procedures in advanced replication.	60 Unit: 1 second	RW/Startup
RConnectPoolSize	Number of remote connections in the connection pool. These are the connections that are used to execute the remote procedure calls. For performance reasons, we can keep the connections open in the pool for a specified time. If the pool becomes full, and there is call for a node that doesn't exist in the pool, then that call is blocked until there is room in the pool. Valid values range from 1-1000 This parameter is associated with server-maintained remote connections used to execute Remote Stored Procedures in advanced replication.	10	RW/Startup
RConnectRPCTimeout	RPC timeout for remote connections. Default is 0 (no timeout). This parameter is associated with server-maintained remote connections used to execute Remote Stored Procedures in advanced replication.	0. Unit 1 millisecond	RW/Startup
ReadBufSize	Sets the buffer size in bytes for the data read from the network	OS dependent	RW/Startup
SocketLinger	This parameter controls the TCP socket option SO_LINGER. It indicates if the system attempts to deliver any buffered data (Yes), or if the system discards it (No), when a close() is issued. The parameter affects all server side connections, including advanced replication and HotStandby.	Yes	RW/Startup

Table 37. Communication parameters (continued)

[Com]	Description	Factory Value	Access Mode
SocketLingerTime	This parameter defines the length of the time interval (in seconds) the socket lingers after a close is issued. If the time interval expires before the graceful shutdown sequence completes, an abortive shutdown sequence occurs (the data is discarded). The default value zero indicates that the system default is used (typically, 1 second)	0	RW/Startup
TcpKeepAlive	This parameter can only be used for Linux, HP-UX, Solaris and QNX platforms. On other platforms, the parameter has no effect. If the client computer is rebooted, the connection status on the server side remains 'ESTABLISHED'. You can set the SO_KEEPALIVE socket option with this parameter. See also parameters TcpKeepAliveIdleTime, TcpKeepAliveProbeCount and TcpKeepAliveProbeInterval.	No	RW/Startup
TcpKeepAliveIdleTime	This parameter can only be used for Linux, HP-UX, Solaris and QNX platforms. On other platforms, the parameter has no effect. This parameter controls the TCP_KEEPIDLE socket option. If the SO_KEEPALIVE option is enabled with the TcpKeepAlive parameter, TCP sends a keepalive probe to the remote system of a connection that has been idle for a period of time. If the remote system does not respond to the keepalive probe, TCP retransmits a keepalive probe for a certain number of times before a connection is considered to be broken. TCP_KEEPIDLE specifies the number of seconds before TCP will send the initial keepalive probe. See also parameters TcpKeepAlive, TcpKeepAliveProbeCount and TcpKeepAliveProbeInterval.	7200	RW/Startup

Table 37. Communication parameters (continued)

[Com]	Description	Factory Value	Access Mode
TcpKeepAliveProbeCount	This parameter can only be used for Linux, HP-UX, Solaris and QNX platforms. On other platforms, the parameter has no effect. This parameter controls the TCP_KEEPCNT socket option. If the SO_KEEPALIVE option is enabled with the TcpKeepAlive parameter, TCP sends a keepalive probe to the remote system of a connection that has been idle for a period of time. If the remote system does not respond to the keepalive probe, TCP retransmits a keepalive probe for a certain number of times before a connection is considered to be broken. The TCP_KEEPCNT option specifies the maximum number of keepalive probes to be sent. See also parameters TcpKeepAlive, TcpKeepAliveIdleTime and TcpKeepAliveProbeInterval.	9	RW/Startup
TcpKeepAliveProbeInterval	This parameter can only be used for Linux, HP-UX, Solaris and QNX platforms. On other platforms, the parameter has no effect. This parameter controls the TCP_KEEPINTVL socket option. If the SO_KEEPALIVE option is enabled with the TcpKeepAlive parameter, TCP sends a keepalive probe to the remote system of a connection that has been idle for a period of time. If the remote system does not respond to the keepalive probe, TCP retransmits a keepalive probe for a certain number of times before a connection is considered to be broken. The TCP_KEEPINTVL option specifies the number of seconds to wait before retransmitting a keepalive probe. See also parameters TcpKeepAlive, TcpKeepAliveIdleTime and TcpKeepAliveProbeCount.	75	RW/Startup

Table 37. Communication parameters (continued)

[Com]	Description	Factory Value	Access Mode
Trace	If this parameter is set to yes, trace information on network messages for the established network connection is written to a file specified with the TraceFile parameter. The factory value for the TraceFile parameter is `soltrace.out`.	no	RW/Startup
TraceFile	If the Trace parameter is set to yes, trace information on network messages is written to a file specified with this TraceFile parameter.	`soltrace.out` (written to the current working directory of the server or client depending on which end the tracing is started)	RW/Startup
WriteBufSize	Sets the buffer size in bytes for the data written into the network	OS dependent	RW/Startup

General section

Table 38. General parameters

[General]	Description	Factory Value	Access Mode
BackupBlockSize	Block size for backup file writing	64 KB Unit: 1 byte k=KB	RW/Startup
BackupCopyIniFile	If set to yes, `solid.ini` file will be copied to the backup directory	yes	RW/Startup
BackupCopyLog	If set to yes, backup operation will copy log files to the backup directory	yes	RW/Startup
BackupCopySolmsgOut	If set to yes, `solmsg.out` file is copied to the backup directory	yes	RW/Startup
BackupDeleteLog	If set to yes, old log files will be deleted after backup operation	yes	RW/Startup

Table 38. General parameters (continued)

[General]	Description	Factory Value	Access Mode
BackupDirectory	Makes a backup of the database, log files, and the configuration file `solid.ini`, using the factory value 'backup' or a given name. For example, BackupDirectory=abc, creates a backup on directory 'abc'. The backup directory must exist and it must have enough disk space for the backup files. It can be set to any existing directory, except the solidDB database file directory, the log file directory, or the working directory. All directory definitions are relative to the solidDB working directory unless the full path is provided. Note that the backup directory entry must be a valid path name in the server's operating system. For example, if the server runs on a UNIX operating system, path separators must be slashes instead of backslashes.	'backup' directory	RW/Startup
BackupStepsToSkip	Controls how frequently netcopy and backup tasks are executed. The value is a number of the tasking system steps that are skipped between backup execution phases. Reasonable values are in the range of 2 - 20. With the factory value 0, the backup proceeds with the maximum speed.	0 (no skipping)	RW/Startup

Table 38. General parameters (continued)

[General]	Description	Factory Value	Access Mode
CheckpointDeleteLog	If this parameter is set to yes, then the server deletes the transaction log file(s) after each successful checkpoint. This saves disk space, but makes it impossible to recover data by rolling forward the logs. The transaction logs contain a copy of the transactions executed by the server. If the database file is erased or corrupted, and if you have kept the transaction log files, then you can restore the data by restoring the backup database file and then rolling forward all the transaction logs that accumulated since the last backup. If you deleted those transaction logs, then you will lose all transactions since the last successful backup. You should only set CheckpointDeleteLog to yes if your database has data that you are willing to risk losing (e.g. test data created during development). See also the BackupDeleteLog parameter. NOTE: If you are using HotStandby and if you set CheckpointDeleteLog=Yes on the Primary server, then the server deletes only the logs that are already acknowledged by Secondary. For example, if the Secondary is down and the Primary is in PRIMARY ALONE state, then the Primary will keep the logs even after the data has been checkpointed on the Primary.	no	RW/Startup
CheckpointInterval	The number of writes to the log files made in the database which causes automatic checkpoint creation. A large setting can delay checkpoints and make them larger. A small setting will guarantee a small checkpoint size. SEE ALSO: MinCheckpointTime. (Note that CheckpointInterval and MinCheckpointTime use different units of measurement. CheckpointInterva is based on the number of log writes, while MinCheckpointTime specifies the minimum time between consecutive checkpoints.)	50000 log writes	RW
DataDictionaryErrorMaxWait	When a data "dictionary operation active" error for prepared statements occurs, the server automatically attempts to reprepare the SQL statement, for the time specified with this parameter. If the table is still compatible with the SQL statement, the operation can continue without any errors reported to the user. This parameter should only be enabled when the thread/client mode is used (Srv.ReadThreadMode=2), because the wait blocks the waiting thread.	0 (Disabled) Unit: 1 second	RW/Startup
DecimalPrecAsNumeric	If set to "yes", the precision of NUMERIC is allowed to be greater than specified.	No	

Table 38. General parameters (continued)

[General]	Description	Factory Value	Access Mode
DefaultStoreIsMemory	If set to Yes, then new tables are created as in-memory tables if they are created without an explicit STORE clause in the CREATE TABLE statement. If set to No, then by default new tables are stored on disk. You can override the factory value by using the STORE clause in the CREATE TABLE statement. Note that system tables are stored on disk, even if this parameter is set to Yes.	Yes	RW
DisableIdleMerge	If set to yes, database is set to disable idlemerge.	No	RW/Startup
FileWriteFlushMode	filewriteflushmode=0 means no flushing after write or read operations. filewriteflushmode=1 means flush before reading from the file. filewriteflushmode=2 means flush after write operations (recommended for vxworks)	0 on most platforms.	RW/Startup
IOThreads	Number of helper I/O threads (per IO device) for read and write purposes. Note: You can restrict the number of write threads with the General.WriterIOThreads parameter. Must be IOThreads > WriterIOThreads. If this rule is violated, the IOThreads parameter takes the precedence (wins).	5	RW/Startup

Table 38. General parameters (continued)

[General]	Description	Factory Value	Access Mode
LockHashSize	The server uses a hash table (array) to store lock information. If the size of the array is remarkably underestimated the performance degrades. Too large hash table doesn't affect directly to the performance although it causes memory overhead. The LockHashSize determines the number of elements in hash table. This information is needed when the server is using pessimistic concurrency control (i.e. locking). The server uses separate arrays for in-memory tables and disk-based tables. This parameter applies to disk-based tables. In general, the more locks you need, the larger this array should be. However, it is difficult to calculate the number of locks that you need, so you will probably need to experiment to find the best value for your applications. The value that you enter is the number of hash table entries. Each table entry has a size of one pointer (4 bytes in 32-bit architectures). Thus, for example, if you choose a hash table size of 1,000,000, then the amount of memory required is 4,000,000 bytes (assuming 32-bit pointers).	1000	RW/Startup

Table 38. General parameters (continued)

[General]	Description	Factory Value	Access Mode

LockWaitTimeOut	LockWaitTimeout specifies the time in seconds that the engine waits for a lock to be released. When the timeout interval is reached, solidDB terminates the timed-out transaction.
	
	For example, if one user is querying a specific row in a table and a second user is updating the same row, the second user's update will wait until either the first user's query is completed or the second user times out. If the first user's query is completed before the second user times out, then the second user is issued a lock for the update.
	
	The maximum lock timeout is 1000 seconds. The server won't start if the default lock timeout in `solid.ini` is more than 1000 seconds.
	
	Note: You can set the lock time out for a single connection by using the following SQL command:
	`SET LOCK TIMEOUT timeout_in_se`
	
	You can change the granularity of the SET LOCK TIMEOUT command from seconds to milliseconds by appending "MS" to the number, e.g.
	`SET LOCK TIMEOUT 500MS`
	
	Note: The SET LOCK TIMEOUT command does not change the setting in the `solid.ini` file.
	
	See also TableLockWaitTimeOut.
		30
		
		Unit: seconds
			RW

LongSequential SearchLimit	Sets the number of sequential fetches after which search is treated as long sequential search	500	
MaxMergeParts	This parameter is used to specify the maximum number of concurrent merge operations, or the number of merge parts.	100	RW/Startup
MaxMergeTasks	The merge process can use multiple merge tasks to accelerate the cleaning up of Bonsai Tree. This parameter specifies the maximum number of merge tasks.	5	RW/Startup
MaxOpenFiles	Sets the maximum number of files kept concurrently open during solidDB sessions	OS dependent	RW/Startup

Table 38. General parameters (continued)

[General]	Description	Factory Value	Access Mode
MaxWriteConcurrency	Limits the number of concurrent row writes (update/deletes/insert) performed at a time. The optimal value depends on the number of available cores (CPUs) and the scattering of updates among different tables. The more cores are available and the more the writes are scattered, the higher is the optimal value. The value should not be higher than the number of available cores (CPUs). Value 0 means there is no limit on the number of concurrent writes.	4	RW/Startup
MergeInterval	Sets the number of index inserts made in the database that causes the merge process to start	Cache size dependent	RW
MinCheckpointTime	Specifies the minimum time in seconds between two checkpoint operations. SEE ALSO: CheckpointInterval. (Note that CheckpointInterval and MinCheckpointTime use different units of measurement. CheckpointInterval is based on the number of log writes, while MinCheckpointTime specifies the minimum time between consecutive checkpoints.)	300 Unit: 1 second	RW
MinMergeTime	This sets a minimum time (in seconds) between two merge operations. For more information about merge operations, see the sections "solidDB Bonsai Tree multiversioning and concurrency control" on page 4 and "Setting the MergeInterval parameter" on page 92.	0	RW
NetBackupConnect	This sets the connect string to the Netbackup Server.	No factory value.	RW/Startup
NetBackupConnectTimeout	Sets the maximum time in milliseconds that a netbackup operation waits for a connection to a NetBackup Server.	For example, to set the timeout to 30 seconds use value 30000 (milliseconds). 0 (no timeout)	RW/Startup
NetBackupCopyIniFile	If set to "yes" the `solid.ini` configuration file is copied to the remote backup directory.	Yes	RW/Startup
NetBackupCopyLog	If set to "yes" the log files are copied to the remote backup directory.	Yes	RW/Startup
NetBackupCopy SolmsgOut	If set to "yes" the `solmsg.out` message file is copied to the remote backup directory.	Yes	RW/Startup
NetBackupDeleteLog	If set to "yes" the backed-up log files are deleted from the source server after the NetBackup has accomplished.	Yes	RW/Startup

Table 38. General parameters (continued)

[General]	Description	Factory Value	Access Mode
NetBackupDirectory	Sets the remote backup directory. The path expression may be relative or absolute. Non-absolute paths are related to the working directory of the NetBackup Server.	No factory value.	RW/Startup
NetBackupReadTimeout	Sets the maximum time in milliseconds that any operation waits for the response from the NetBackup Server. For example, to set the timeout to 30 seconds use value 30000 (milliseconds).	60 000	RW/Startup
Pessimistic	When you specify PESSIMISTIC concurrency control, the server places locks on rows to control the level of consistency and concurrency when users are submitting queries or updates to rows. The factory value is 'No', that is, the server uses optimistic concurrency control. However, by setting the Pessimistic parameter to 'Yes', you can tell the server to default to pessimistic locking for any new tables that are created AND for any old tables whose concurrency control method was never explicitly set with the ALTER TABLE command. If you set a table's locking mode by using the command `ALTER TABLE base_table_name SET {OPTIMISTIC | PESSIMISTIC}` the ALTER TABLE command takes precedence. For a detailed explanation of pessimistic vs. optimistic concurrency control, as well as a discussion of whether the Pessimistic parameter in `solid.ini` takes precedence over other methods of setting the concurrency control, see *IBM solidDB SQL Guide*.	No	RW/Startup
ReadLevelMaxTime	This parameter specifies in seconds how long an SQL execute can hold the transaction read level in the READ COMMITTED isolation level until it is released. The default value is 10 seconds.	10	RW/Startup
Readonly	if set to yes, database is set to read-only mode	No	RW/Startup

Table 38. General parameters (continued)

[General]	Description	Factory Value	Access Mode
SearchBufferLimit	Sets the maximum percentage of search buffers from the total buffered memory reserved for open cursors. The search buffer contains a local copy of the last B-tree page. Because of this, active searches do not need to go through the index and cache manager to get to the next row in the search. Instead, the searches read the local copy residing in the cache manager. Other searches can also access the page for read-only purposes unless it has been modified by a transaction. When calculating the buffer limit value, take the approximate number of active searches in the database and multiply it by two. The result is the need for search buffers. After this, you can calculate the suitable percentage from the cache size.	50	RW/Startup
StartupForceMerge	If this parameter is set to Yes, it forces a merge operation to run when the server is started. The server accepts no user commands until the merge operation has been completed.	No	RW/Startup

TableLockWaitTimeout	Sets the time in seconds that a transaction waits to get a lock. When messages are executed in the replica, it is possible to run them in pessimistic or mixed concurrency mode, which means table level locks are used.
	
	There are times when a transaction will acquire an exclusive lock to a table. If there is a conflict, this setting provides the transaction's wait period until the exclusive or shared lock is released. This parameter is used for synchronized databases only.
	
	Table level locks are used when the PESSIMISTIC keyword is explicitly provided in the following solidDB commands:
	`IMPORT SUBSCRIPTION`
	`MESSAGE` *`message_name`* `EXECUTE`
	`(only with NO EXECUTE option)`
	`MESSAGE` *`message_name`* `FORWARD`
	`MESSAGE` *`message_name`* `GET REPLY`
	`DROP SUBSCRIPTION`
	
	See also LockWaitTimeOut.
		30
		
		Unit: 1 second
			RW

TransactionEarlyValidate	Transaction early validating is used when this parameter is set to yes. The possible values are yes and no.	Yes	RW/Startup

Table 38. General parameters (continued)

[General]	Description	Factory Value	Access Mode
TransactionHashSize	The hash table contains slots that are occupied by incomplete (i.e. open) transactions. The transaction hash size sets the size of the table for open transactions. Once the number of occupied slots increases, the operations with this table become slower. The database offers higher performance when the average number of transactions per slot is lower (5 is a good initial limit for the transaction per slot average). Note that you can monitor the status of this hash table using ADMIN COMMAND 'report *filename*'. For example: `ADMIN COMMAND 'report myfile.txt'` The output contains the following related information: tablesize = setting nused = slots taken from hash table list length = sum of all transactions in the table	4000, but depends partly on the cache size. Minimum value is 1000. Maximum is 50,000.	RW/Startup
VersionedPessimisticReadCommitted	If this parameter is enabled, pessimistic D-tables use versioned reads with READ COMMITTED isolation. Read with FOR UPDATE work as before. In other words, pessimistic D-tables work like M-tables.	Yes	RW/Startup
VersionedPessimisticRepeatableRead	If this parameter is enabled, pessimistic D-tables use versioned reads with REPEATABLE READ isolation.	Yes	RW/Startup
WriterIOThreads	Number of helper threads dedicated to writing tasks (per IO device). Must be IOThreads > WriterIOThreads. If this rule is violated, the factory value is used. If IOThreads=1 then the setting WriterIOThrreads=0 is enforced.	1	RW/Startup

HotStandby section

Table 39. HotStandby parameters

Parameter name	Description	Factory value	Access mode
1SafeMaxDelay	In 1-Safe replication, the maximum delay before a committed transaction is sent to the Secondary (in milliseconds).	5000	RW

Table 39. HotStandby parameters (continued)

Parameter name	Description	Factory value	Access mode
2SafeAckPolicy	This specifies the timing of the Secondary's acknowledgement when it receives a transaction from the Primary. Valid values are: • 1 = 2-safe received. The Secondary server acknowledges when it receives the data. • 2 = 2-safe visible. The Secondary server acknowledges when the data is "visible", that is, when the Secondary has executed the transaction. • 3 = 2-safe durable. The Secondary server acknowledges when it has made the data durable, that is, when it has committed the data and written the data to the disk. Not surprisingly, 2-safe durable is the safest approach, and 2-safe received has the fastest response time. However, in practice, the 2-safe received mode provides in most cases sufficient guarantees for data safety hence providing the best compromise between safety and speed. This parameter applies only if the server is using 2-safe replication. **Note:** Although this parameter controls the Secondary server's behavior, this parameter is set on the Primary. The value in the Secondary's `solid.ini` value is ignored.	1	RW
AutoPrimaryAlone	If this parameter is set to Yes, then the server is automatically put in PRIMARY ALONE state (rather than PRIMARY UNCERTAIN state) when the connection to the Secondary is broken. If you plan to set this to "yes", read the warnings in Network partitions and dual primaries.	No	RW
CatchupSpeedRate	While the server is performing catchup, it also continues to service database requests from clients. You may use the **CatchupSpeedRate** parameter to give greater importance to responding to application requests and lower priority to catchup, or vice versa. The speed rate is expressed as a percentage of the maximum available speed dictated by the link and Secondary throughput. Larger numbers mean more emphasis on catchup and less on servicing client requests. Allowed values are 1-99.	50	RW

Table 39. HotStandby parameters (continued)

Parameter name	Description	Factory value	Access mode
Connect	The **Connect** parameter indicates the address of the other HotStandby server in the pair. The format of the Connect string in the HotStandby section is the same as the format of the **Listen** parameter in the `[Com]` section. If you omit this parameter in a server that you intend for HotStandby, then you can set this parameter dynamically by using an ADMIN COMMAND. Until the server has a Connect string, the server can only be in the states that do not involve a connection, that is, PRIMARY ALONE, SECONDARY ALONE, and STANDALONE. The **Connect** parameter is ignored unless the **HSBEnabled** parameter is set to "yes". For Transparent Connectivity (TC) connections, the **Connect** parameter can be overridden with the **TCConnect** parameter.	No factory value.	RW

ConnectTimeout	By specifying a connect timeout value, you can set the maximum time in seconds that a HotStandby connect operation waits for a connection to a remote machine.
	
	The **ConnectTimeout** parameter (which is useful only on certain platforms) is only used with certain administration commands. These are:
	• hotstandby connect
	• hotstandby switch primary
	• hotstandby switch secondary
	
	For example, to set the timeout to 30 seconds (30000 milliseconds):
	`[HotStandby]`
	`ConnectTimeout=30000`
	
	See also **PingTimeout**.
		0 (no timeout)
		
		Unit: 1 ms
			RW

CopyDirectory	The **CopyDirectory** parameter in the `[HotStandby]` section defines a name and location for the HotStandby copy operation that is performed when the user executes the command:
	`ADMIN COMMAND 'hotstandby copy';`
	
	For example, the parameter may look like:
	`[HotStandby]`
	`CopyDirectory=C:\solidDB\secondary\dbfiles`
	
	If you provide a relative path for the **CopyDirectory** parameter, the path will be relative to the directory that holds the Primary server's `solid.ini` file.
	
	This parameter has no factory value, so if the directory is not specified in the `solid.ini` file, it must be provided in the copy command.
	
	Please note that ADMIN COMMAND 'hotstandby netcopy' as the more flexible solution is the recommended way to copy the database.
		No factory value	RW

Table 39. HotStandby parameters (continued)

Parameter name	Description	Factory value	Access mode
HSBEnabled	If this parameter is set to yes, the server may act as a HotStandby Primary or Secondary server. If this parameter is set to no, then the server may not act as a HotStandby server. Setting this parameter to Yes will implicitly define the default initial state of the server to be SECONDARY ALONE when the server first starts. Valid values are "yes" and "no". To use HotStandby, you must also specify the **Connect** parameter, either by setting it in the `solid.ini` file or by using an ADMIN COMMAND to set it.	no	RO
MaxLogSize	Maximum size of the disk-based HSB log. The factory value: unlimited	0 Unit: 1 byte k=KB m=MB	
MaxMemLogSize	When the file-based logging is disabled (**Logging.LogEnabled**=No), the size of the in-memory log holding transactions before they are sent to the Secondary. The value affects the time the server may stay in the PRIMARY ALONE state, before the in-memory log becomes full.	8M Unit: 1 byte k=KB m=MB	RO
NetcopyRpcTimeout	Data transmission acknowledgment timeout for netcopy operation (in milliseconds)	30000 Unit: 1 ms	RW
PingInterval	The Primary and Secondary send "ping" messages to each other at regular intervals to make sure that they are still connected. (These pings are independent of the transaction information that the Primary sends to the Secondary.) The value is equal to the interval (in milliseconds) between two consecutive pings sent by a server.	1000 (one second) Unit: 1 ms	RW
PingTimeout	The parameter specifies how long a server should wait before concluding that the other server is down or inaccessible. After the time specified (in milliseconds) has passed the server concludes that a connection is broken and changes the state accordingly. See also **ConnectTimeout**.	4000 (four seconds) Unit: 1 ms	RW
PrimaryAlone	This parameter is deprecated. Use the **AutoPrimaryAlone** parameter.	No	RW
SafenessLevel	This parameter sets the safeness level of the replication protocol. By using the "auto" value, you can allow the safeness level to dynamically change in relation to the durability level. If you set **SafenessLevel** to "auto" and set the durability to relaxed by using the SET DURABILITY command or the **DurabilityLevel** parameter, the safeness level is set to 1-safe, and when you set the durability level to strict, the safeness level is set to 2-safe. However, if **DurabilityLevel** is set to 2 (Adaptive Durability), the "auto" setting has no effect - the safeness level will always be 2-safe.	Possible values are: 1safe, 2safe and auto	RW

Table 39. HotStandby parameters (continued)

Parameter name	Description	Factory value	Access mode
TCConnect	This parameter overrides the connect string defined with the **Connect** parameter for the purposes of Transparent Connectivity (TC) connections, where the servers need to use different networks to connect to each other. By default, the secondary servers provide the **Connect** connect string to the TC clients for specifying the location of the primary server. If the servers use different network to connect to each other and TC clients cannot or are not supposed to use the same network, the TCConnect parameter can be used to override the **Connect** connect string.	No factory value.	RW

IndexFile section

Table 40. IndexFile parameters

[IndexFile]	Description	Factory Value	Access Mode
BlockSize	Sets the block size of the database file in bytes; use multiple of 2 KB: minimum 2 KB, maximum 64 KB	16 KB Unit: 1 byte k=KB	RO
CacheSize	Sets the size of database cache memory for the server in bytes; the minimum is 512 kilobytes. Although solidDB is able to run with a small cache size, a larger cache size speeds up the server. The cache size needed depends on the size of the database file, the number of connected users, and the nature of the operations executed against the server. You can change the CacheSize value dynamically as follows: admin command 'parameter IndexFile.CacheSize=40mb' **Attention:** Setting the CacheSize to a value larger than the amount of memory available may significantly degrade performance. If your system only has a small amount of free memory available, you should reduce the CacheSize.	32 MB Unit: 1 byte k=KB m=MB	RW
DirectIO	Defines if the index file uses Direct I/O. Direct I/O means that operating system buffer pool is bypassed in file I/O. This parameter is not effective in Windows environments; in Windows environments, database files always use Direct I/O.	No	RW/Startup
ExtendIncrement	Sets the number of blocks of disk space that are allocated at one time when solidDB needs to allocate more space for the database file. Currently, each block is 8KB. E.g. a value of 500 (8KB blocks) corresponds to 4 MB of disk space.	500	RW/Startup

Table 40. IndexFile parameters (continued)

[IndexFile]	Description	Factory Value	Access Mode
FileSpec_[1... N]	Defines the location and the maximum size of the index file. Note that in solidDB, the term "index file" is used as a synonym for "database file." The parameter accepts the following three arguments: database file name followed with maximum size (in bytes) of the database file, for example: `FileSpec_1=c:\soldb\solid.db 200000000` This parameter also has an optional argument after the maximum size: device number, which is the physical drive number. The number value itself is not essential, but it is used as a hint for I/O threads, allowing the server to perform database file I/O requests in a parallel manner if you split the file into multiple physical disks. The *N* in the parameter syntax signifies the number of the file if the database file is divided into multiple files and onto multiple disks. For details, read “FileSpec_[1...n] parameter” on page 50. To achieve better performance, the database file must be stored to a local drive using local disk names to avoid problems with network I/O. Note that you may also want to have multiple files on a single disk if your physical disk is partitioned into multiple logical disks and no single logical disk can accommodate the size of the database file you expect to create.	solid.db 2147483647 (2G-1 bytes)	RW/Startup
PreFlushPercent	Sets the percentage of page buffer which is kept clean by the preflush thread. Note that the preflush operations prepare the cache for the allocation of new blocks. The blocks are written onto the disk from the tail of the cache based on a Least Recently Used (LRU) algorithm. Therefore, when the new cache blocks are needed, they can be taken immediately without writing the old contents onto the disk.	1	RW/Startup
ReadAhead	Sets the number of prefetched index reads during long sequential searches. Note that when the I/O manager is handling a long sequential search, it enters a read-ahead operation mode. This mode ensures that the next file blocks of the search in question are read into the cache in advance. This naturally improves the overall performance of sequential searches.	4	RW/Startup

Table 40. IndexFile parameters (continued)

[IndexFile]	Description	Factory Value	Access Mode
ReferenceCacheSizeForHash	solidDB uses a hash table to ease access to the cache. The hash table size equals the number of pages in the cache. This guarantees almost collision-free access. If the cache size is increased dynamically, the hash table is not automatically enlarged. This results in a higher collision probability. To avoid this, you can use the ReferenceCacheSizeForHash parameter to accommodate the enlarged cache. The ReferenceCacheSizeForHash parameter value is used for calculating the cache hash table size. You should only use the parameter if you know, in advance, what will be the maximum cache size during the server lifecycle. On the other hand, if the value is not given, hash table collisions may occur when the cache size is increased. **Note:** The ReferenceCacheSizeForHash parameter value must not be smaller than the CacheSize value. If it is, the ReferenceCacheSizeForHash parameter value is rejected and the default value is used. Also, a message is printed to the `solmsg.out` log file.	0	RW/Startup
SynchronizedWrite	On UNIX/Linux platforms this parameter may be set to "no" to enact asynchronous I/0. Asynchronous I/O provides, in general, more performance but it can cause higher variance of response latencies (lower latency determinism).	yes	RO

Logging section

Table 41. Logging parameters

[Logging]	Description	Factory Value	Access Mode
BlockSize	Sets the block size of log files. The log block size may be changed between startups. Logs having block size different than the one set are accepted at recovery. The value has to be a multiplicity of 1 KB. Bigger blocks reduce the overhead of log writing.	16 KB Unit: byte k=KB	RW/Startup
DigitTemplateChar	Specifies the template character that will be replaced in the name template of the log file. See the description of the FileNameTemplate for more details.	#	RW/Startup
DirectIO	Defines if the log file uses Direct I/O. Direct I/O means that operating system buffer pool is bypassed in file I/O. This parameter is not effective in Windows environments; in Windows environments, database files always use Direct I/O.	No	RW/Startup

Table 41. Logging parameters (continued)

[Logging]	Description	Factory Value	Access Mode

DurabilityLevel	This parameter controls whether the transaction durability level is "strict", "relaxed", or "adaptive". If durability is "strict", then writes to the transaction log are synchronous — i.e. as soon as a transaction has been committed, the transaction is written to the transaction log. If durability is "relaxed", then writes are asynchronous — there may be a delay between the time that the transaction is committed and the time that it is logged. For a detailed explanation of "strict" and "relaxed" durability, see “Logging and transaction durability” on page 83.
	
	The possible values are:
	
	1 = relaxed durability
	
	2 = adaptive durability. This value applies only to HSB (HotStandby) Primary servers.
	
	3 = strict durability
	
	The server's durability level may be set dynamically by using the command:
	`ADMIN COMMAND 'parameter Logging.DurabilityLevel=n';`
	
	where n is one of the valid values for this parameter.
	
	Each connection may override this `solid.ini` parameter by using the SET DURABILITY or SET TRANSACTION DURABILITY command. See chapter "SET" in *solidDB SQL Guide*.
	
	Note that the DurabilityLevel parameter affects the server's behavior only if transaction logging is turned on. If you turn off transaction logging by setting
	`[Logging] LogEnabled=No`
	
	then your data will not be logged to disk, regardless of the setting of DurabilityLevel. If LogEnabled is set to No and DurabilityLevel is set, then the server will briefly display a warning message at the time that it starts.
	
	DurabilityLevel is not the only configuration parameter that influences how the server writes information to logs. You may also want to read about the LogWriteMode parameter, which also offers some options that allow you to trade off speed and reliability. If you are using HotStandby, you may also want to read about the 2SafeAckPolicy parameter.
		1	RW

Table 41. Logging parameters (continued)

[Logging]	Description	Factory Value	Access Mode
FileFlush	This parameter controls log file flush behavior. This parameter is only valid for platforms supporting Synchronized I/O Data Integrity Completion. These are such as Solaris, HP-UX, and Linux. When set to no in these platforms, the operating system, rather than the solidDB engine, flushes the log file.	yes	RW/Startup

FileNameTemplate	Defines the path and naming convention used when creating log files. These log files contain information used to recover data if the server crashes.
	
	To be more specific, this parameter defines at least the naming convention used when creating log files, but not necessarily the path. If this is the case, the Logging.LogDir parameter defines the path. For more information, see the LogDir parameter description.
	
	Template characters (e.g. "#") are replaced with sequential numbers; for example, the following file entry instructs solidDB to create log files in directory `C:\soliddb\log` and to name them sequentially starting from `sol00001.log`.
	`FileNameTemplate = c:\soliddb\log\sol#####.log`
	
	Your template may use between 4 and 10 template characters. If you do not want to use the "#" sign as a template character, you may specify a different character by setting the parameter DigitTemplateChar.
	
	If the number of log files would exceed the maximum possible number (e.g. all names from `sol00001.log` to `sol99999.log` are used up), then the server will give an error message and exit. The error message will be similar to the following:
	`"Error: Illegal log file name template. Most likely the log file name template specified in solid.ini ... contains too few or too many sequence number digit positions. There should be at least 4 and at most 10 digit positions."`
	
	To achieve better performance, the log files must be stored to a local drive using local disk names to avoid problems with network I/O.
		sol#####.log	RW/Startup

Table 41. Logging parameters (continued)

[Logging]	Description	Factory Value	Access Mode
LogDir	This parameter sets the directory prefix of the log file path specified by using the Logging. FileNameTemplate parameter. Effectively, it specifies the log file directory if FileNameTemplate only specifies the file name (default). The default value is the server working directory. The specified directory has to exist prior to starting the server.	"." (the server's working directory)	RW/Startup
LogEnabled	Specifies whether transaction logging is enabled or not. If transaction logging is disabled, you will get better performance but lower transaction durability (if solidDB shuts down unexpectedly, then you lose any transactions since the last checkpoint). Note that this parameter applies to in-memory tables as well as disk-based tables.	yes	RW/Startup
LogWriteMode	Specifies the mode in which the log will be written. The following two modes are available: • 0: ping-pong method • 2: Overwrite method (factory value) The choice of logging method depends on the log file media and the level of security required. For details on each of these methods, read "Transaction logging" on page 37.	2 (Overwrite method)	RW/Startup
MinSplitSize	When this file size is reached, logging will be continued to the following log file after the next checkpoint	10 MB Unit: 1 KB k=KB m=MB	RW/Startup
RelaxedMaxDelay	This sets the maximum time in milliseconds that the server waits until the committed transaction(s) are written to the log. This parameter applies only when the transaction durability level is set to RELAXED (with the DurabilityLevel parameter or the SET DURABILITY statement). The units are milliseconds. Minimum allowed value: 100 (i.e. 100 milliseconds).	5000 milliseconds (5 seconds) Unit: 1 ms	RW/Startup
SyncWrite	This parameter applies only to platforms, such as Solaris, HP-UX, and Linux, which support Synchronized I/O Data Integrity Completion. When set to yes, solidDB assumes that the platform supports Synchronized I/O Data Integrity Completion. It should be set to No on all other platforms.	no	RW/Startup

LogReader section

Table 42. LogReader parameters

[LogReader]	Description	Factory Value	Access Mode
LogReaderEnabled	By using this parameter, you can enable or disable the log reader capability. In solidDB Universal Cache and configurations with InfoSphere™ CDC Replication, this parameter must be set to Yes.	no	RO
MaxLogSize	This parameter defines the size of the protected portion of the disk-based transaction log. When the log files are removed, for example, in conjunction with a backup, at least the specified amount of the log data is retained. The protected portion of the log facilitates a possible catchup after a failure case when the replication has not been active for some time. The actual log size may exceed the **MaxLogSize** value, if the log files are not removed. Catchup is possible as long as the propagator log position is within the existing log. The minimum value is 5 (5 MB). If you attempt to define a smaller log size, it is automatically changed to 5 MB. The maximum possible log size is practically unlimited. Unit: megabytes.	10240	RW

Table 42. LogReader parameters (continued)

[LogReader]	Description	Factory Value	Access Mode
MaxSpace	This parameter defines the maximum number of log records buffered before slowdown. The log records are buffered in an in-memory log reader buffer. The size of a log record is that of the (binary) row size, plus a few bytes of additional metadata overhead. When the buffer fills up, throughput throttling is applied in solidDB: the operations are blocked until there is room in the logreader buffer. The throttling only takes place when the log reading is active. If there is no log reader activity, solidDB continues the processing and log files are preserved at least until the defined **MaxLogSize** limit is reached (see above).	100000	RW
Silent	If set to 'Yes', the Log Reader activities are not output to `solmsg.out`. Possible values are 'Yes' and 'No'.	No	RW/Startup

MME section

Note: The **DefaultStoreIsMemory** parameter (in the [General] section of the `solid.ini` file) is also related to solidDB in-memory database. For more information, see "General section" on page 127.

Table 43. MME parameters

[MME]	Description	Factory Value	Access Mode

ImdbMemoryLimit	This sets an upper limit on the amount of memory (virtual memory) that the server will allocate for in-memory tables and indexes on in-memory tables. Note that "in-memory tables" includes Temporary Tables and Transient Tables, as well as "normal" (persistent) in-memory tables.
	
	The limit may be specified in bytes, kilobytes (kb), megabytes (mb), or gigabytes (gb). For example:
	`ImdbMemoryLimit=1073741824`
	`ImdbMemoryLimit=1048576kb`
	`ImdbMemoryLimit=1024MB`
	`ImdbMemoryLimit=1GB`
	
	If you use the value 0, it means "no limit".
	
	As a general rule, for servers with 1 GB or less of memory, the maximum amount that you should allocate to in-memory tables is usually 30% - 70% of the system's physical memory. The more memory the system has, the larger the percentage of it you may use for in-memory tables.
	
	For more details about controlling memory usage of in-memory tables, see *IBM solidDB In-Memory Database User Guide*.
	
	Note: This parameter only applies only to solidDB main memory engine tables. It does not apply to other versions of solidDB or to disk-based tables.
	
	You can change this with the command:
	`ADMIN COMMAND 'parameter MME.ImdbMemoryLimit=n[kb|mb|gb]';`
	
	where 'n' is a positive integer. You may only increase, not decrease, this value while the server is running. The command takes effect immediately. The new value is written back to the `solid.ini` file at shutdown.
	
	CAUTION:
	We strongly recommend that you ensure that your in-memory tables will fit within the available physical memory. If you exceed the amount of physical memory available, performance will decrease significantly. If you use up all of the available virtual memory, the server will abruptly limit inserts, updates, etc. and will return error codes.
		0
		
		Unit: 1 byte k=KB m=MB g=GB
			RW

Table 43. MME parameters (continued)

[MME]	Description	Factory Value	Access Mode
ImdbMemoryLowPercentage	Once you have set ImdbMemoryLimit, you may set this additional parameter to give you advance warning before you use up all of memory. This ImdbMemoryLowPercentage parameter allows you to indicate what percentage of memory you may use before the server starts limiting your ability to insert rows into in-memory tables, etc. For example, if ImdbMemoryLimit is 1000MB and ImdbMemoryLowPercentage is 90 (percent), then the server will stop accepting inserts when you've used up 900 megabytes of memory for your in-memory tables. Valid values are between 60 and 99 (percent). For more details about controlling memory usage of in-memory tables, see *IBM solidDB In-Memory Database User Guide*. **Note:** This parameter only applies to solidDB main memory engine tables. It does not apply to other versions of solidDB or to disk-based tables.	90	RW/Startup
ImdbMemoryWarningPercentage	This parameter sets a warning limit for the IMDB memory size. The warning limit is expressed as a percentage of the **ImdbMemoryLimit** parameter value. When the **ImdbMemoryWarningPercentage** limit is exceeded, a system event is given. The **ImdbMemoryWarningPercentage** parameter value is automatically checked for consistency. It must be lower than the **ImdbMemoryLimit** parameter value. For more details about controlling memory usage of in-memory tables, see *IBM solidDB In-Memory Database User Guide*. **Note:** This parameter only applies to solidDB main memory engine tables. It does not apply to other versions of solidDB or to disk-based tables.	90	RW/Startup
LockEscalationEnabled	Typically, when the server needs to use locks to prevent concurrency conflicts, the server locks individual rows. This means that each user affects only those other users who want to use the same row(s). However, the more rows are locked, the more time the server must spend checking for conflicting locks. In some cases, it is worthwhile to lock an entire table rather than a large number of the rows in that table. When LockEscalationEnabled is set to yes, the lock level is escalated from row-level to table-level after a specified number of rows (in the same table) have been locked within the current transaction. Lock escalation improves performance, but reduces concurrency, because it means that other users are temporarily unable to use the same table, even if they want to use different rows within that table. See the parameter LockEscalationLimit. The value may be "yes" or "no". **Note:** This parameter applies to in-memory tables only.	no	RW/Startup

Table 43. MME parameters (continued)

[MME]	Description	Factory Value	Access Mode
LockEscalationLimit	If LockEscalationEnabled is set to yes, then this parameter indicates how many rows must be locked (within a single table) before the server will escalate lock level from row-level to table-level. (See LockEscalationEnabled for more details.) The value may be any number from 1 to 2,147,483,647 (2^32-1). **Note:** This parameter applies to in-memory tables only.	1000	RW/Startup
LockHashSize	The server uses a hash table (array) to store lock information. If the size of the array is remarkably underestimated the performance degrades. Too large hash table doesn't affect directly to the performance although it causes memory overhead. The **LockHashSize** determines the number of elements in hash table. This information is needed when the server is using pessimistic concurrency control (i.e. locking). The server uses separate arrays for in-memory tables and disk-based tables. This parameter applies to in-memory tables. In general, the more locks you need, the larger this array should be. However, it is difficult to calculate the number of locks that you need, so you will probably need to experiment to find the best value for your applications. The value that you enter is the number of hash table entries. Each table entry has a size of one pointer (4 bytes in 32-bit architectures). Thus, for example, if you choose a hash table size of 1,000,000, then the amount of memory required is 4,000,000 bytes (assuming 32-bit pointers).	1000000	RW/Startup
MaxBytesCachedInPrivateMemoryPool	This parameter defines the maximum bytes stored into the free list of MME's private memory pool (private memory pool is private for each main-memory index). If there is more free memory in the private pool, the extra memory is merged into global pools. Value 0 means immediate merge to global pool, usually degrades performance, but minimizes memory footprint. There is no maximum value; the default value of 100000 gives good performance with little memory overhead.	100000	RW/Startup
MaxCacheUsage	The value of **MaxCacheUsage** limits the amount of D-table cache used while checkpointing M-tables. The value is expected to be given in bytes. Regardless of the value of the **MaxCacheUsage** at most half of the D-table cache (**IndexFile.CacheSize**) is used for checkpointing M-tables. Value **MaxCacheUsage=0** sets the value unlimited, which means that the cache usage is **IndexFile.CacheSize/2**.	8MB	RW/Startup

Table 43. MME parameters (continued)

[MME]	Description	Factory Value	Access Mode
NumberOfMemoryPools	This parameter defines the number of global memory pools. Bigger values may give better performance on multicore systems with certain load scenarios but they also increase memory slack and hence server process size. Minimum value is 1. There is no maximum value; however, the number of cores in the system should not be exceeded.	1	RW/Startup
ReleaseMemoryAtShutdown	When set to "yes", this tells the server that when it shuts down it should explicitly release memory used by in-memory tables, rather than relying on the operating system to clean up all memory associated with this process. Some operating systems (like VxWorks) may require you to set this to "yes" to ensure that all memory is released. The possible values are yes and no. The factory value is no because shutting down the server is faster that way.	No	RW/Startup

Sorter section

Table 44. Sorter parameters

[Sorter]	Description	Factory Value	Access Mode
BlockSize	Block size of the external sorter files. With the factory value 0, the database block size is used.	0	RW/Startup
MaxCacheUsePercent	This parameter sets the maximum percentage of cache pages that can used for sorting. The valid values range from 10% to 50%. E.g. if the CacheSize (in the IndexFile section of the `solid.ini` file) is 20MB, and if MaxCacheUsePercent is 25, then a maximum of 5MB of memory is available for sorting. If you specify both the MaxCacheUsePercent and the MaxMemPerSort, the values must be compatible. You get an error message if the following is not true: MaxCacheUsePercent x CacheSize >= MaxMemPerSort	25 (that is, 25 percent)	RW/Startup
MaxFilesTotal	Maximum number of files used for sorting	100	RW/Startup

Table 44. Sorter parameters (continued)

[Sorter]	Description	Factory Value	Access Mode
MaxMemPerSort	This parameter sets the maximum memory available in bytes for one sort (that is, sorting the result set of one query). This value must not exceed the amount of memory available to the sorter (see MaxCacheUsePercent for more information).		RW/Startup
SorterEnabled	This parameter enables or disables the usage of the external sorter.	Yes	RW/Startup

TmpDir_[1... N]	When this parameter is specified in the configuration file, the external sorter algorithm is enabled. The external sorter algorithm is used for sorting processes that do not fit in main memory. The parameter defines the name of the directory (or directories) that contain temporary files created when using the external sorter algorithm. The *N* signifies the file directory number if more than one directory is used to store the temporary file. For example:
	`TmpDir_1=c:\soldb\temp1`
	`TmpDir_2=d:\soldb\temp2`
		Defaults to ".", in other words, the current directory (the directory from which the server was started).	RW/Startup

SQL section

Table 45. SQL parameters

[SQL]	Description	Factory Value	Access Mode
AllowDuplicateIndex	If set to yes, allows duplicate index definitions. This is a backward compatibility parameter. In versions preceding 4.5, it was possible to create duplicate indexes.	no	RO
CharPadding	When set to yes, enforces SQL standard padding of CHAR values with blanks (right-filled) to the length defined for the column. With the default setting (no), the blanks are discarded. The value of the parameter does not affect comparisons (where blanks are always discarded). This feature is not implemented in the solidDB SQL Editor (solsql). Use ODBC3 or JDBC2 drivers with this feature. Notice also that this parameter affects the ODBC and JDBC driver behaviour.	no	RO

Table 45. SQL parameters (continued)

[SQL]	Description	Factory Value	Access Mode
ConvertOrsToUnionsCount	This parameter specifies the maximum number of OR operations that may be converted to UNION operations. Note that this parameter does not force the optimizer to convert OR operations to UNION operations; it merely sets a maximum limit on the number of OR operations that the server may convert to UNION operations.	0	RW/Startup
CursorCloseAtTransEnd	By default, the solidDB ODBC driver closes all the cursors opened from the user connection when a commit is called with SqlTransact from this connection. If this parameter is set to No, the cursors are kept open.	yes	RO
DecFloatPrecision16	If set to 'Yes', the precision of the decimal float data type is limited to 16 (same as in solidDB 4.5 and earlier). In storage, the decimal float type is inflicted by the column type specification 'DECIMAL' (without scale and precision). Also, expressions involving DECIMAL or NUMERIC data types may produce decimal float values. By default (No), the precision of the decimal float data type is 52. Possible values are 'Yes' and 'No'.	No	RO
EmulateOldTimestampDiff	If included in the `solid.ini` file and set to "Yes", the old TIMESTAMPDIFF behavior is emulated by the server. This old behavior returns the integer number of intervals of type interval by which timestamp_exp2 is greater than timestamp_exp1. Otherwise, the default is the new behavior which returns the integer number of interval as the amount of full units between timestamp_exp1 and timestamp_exp2.	"No"	RW/Startup
EnableHints	If this parameter is included in the `solid.ini` file and set to "Yes", hints are enabled. If set to "No," hints are disabled. For details on hints, read "Using Optimizer Hints" in *IBM solidDB SQL Guide*. Sometimes hints in queries may produce undesirable effects. They may be disabled by setting this parameter to "no"	yes	RW/Startup
ExecuteNodataODBC3Behaviour	By default, when the execution of a DELETE or UPDATE statement does not affect any rows, the statement returns SQL_SUCCESS. This is the ODBC v.2 behavior. By setting this parameter to 'yes', the SQLSTATE returned in those cases is SQL_NO_DATA, which conforms with ODBC v.3.	No	RW/Startup

Table 45. SQL parameters (continued)

[SQL]	Description	Factory Value	Access Mode
Info	Sets the level of informational messages [0-8] printed from the server (0=no info, 8=all info); information is written into the file defined by parameter **InfoFileName**, or into `soltrace.out` if **InfoFileName** is not defined.	0	RW/Startup
SQLInfo	Sets the level of informational SQL level messages [0-8] (0=no info, 8=all info); information is written into a file defined by parameter **InfoFileName**, or into `soltrace.out` if **InfoFileName** is not defined.	0	RW/Startup
InfoFileFlush	If set to yes, flushes info file after every write operation	yes	RW/Startup
InfoFileName	Default info file name. The default name is `soltrace.out`. Since the `soltrace.out` file may contain information from several sources, we recommend that you explicitly set **InfoFileName** to another name if you set the Info or SQLInfo parameters to a number larger than 0.	soltrace.out	RW/Startup
InfoFileSize	Sets the maximum size of the info file.	1 MB	RW/Startup
IsolationLevel	Possible values: 3 (SERIALIZABLE) 2 (REPEATABLE READ) 1 (READ COMMITTED) This is the default transaction isolation level. For more information about transaction isolation levels, see the description of the SET TRANSACTION ISOLATION command (part of *IBM solidDB SQL Guide*, Appendix B, solidDB SQL Syntax), and section "Choosing transaction isolation levels" on page 85. In addition to setting this parameter in the `solid.ini` file, you may also set the value by executing the following command: `ADMIN COMMAND 'parameter SQL.IsolationLevel={1 | 2 | 3}'` Note that if you execute this as an admin command, then it takes effect after the server is restarted. Note that in version 4.0 and later, in-memory tables will not work with **IsolationLevel** set to SERIALIZABLE.	1 (Read Committed)	RW/Startup
Latin1CaseSemantics	If set to 'No', uppercase/lowercase conversions are disabled for characters with decimal value between 126 and 256.	Yes	RW/Startup

Table 45. SQL parameters (continued)

[SQL]	Description	Factory Value	Access Mode
MaxBlobExpressionSize	Certain string operations use only the first N bytes of a character value, not the entire value. For example, the LOCATE() operation checks only the first N bytes of the string. If you want to tell the server to check further into (or less far into) long strings, you may set this parameter. By default, the units are kilobytes — e.g. "64" means 64KB You may specify "MB" if you want to express the units in megabytes. This parameter applies to all the character data types, including CHAR, VARCHAR, LONG VARCHARY, WCHAR, WVARCHAR, and LONG WVARCHAR. Since the Wide character data types use 2 bytes per character, the number of characters searched is half the number of bytes. E.g. if you set **MaxBlobExpressionSize** to 64K bytes, then the first 32K characters of Wide character data types will be searched.	1024KB (1MB) Unit: 1 KB m=MB	RW/Startup
MaxNestedProcedures	Sets the maximum number of allowed nested procedures. If this parameter is defined too high, the server stack may become insufficient depending on the operating system.	16	RW/Startup
MaxNestedTriggers	Sets the maximum number of allowed nested triggers. This maximum number includes both direct and indirect nesting, so both A → A → A and A → B → A are counted as three nested triggers.	16	RW/Startup
NumericPadding	If set to Yes, causes output of DECIMAL and NUMERIC to be zero-right-padded up to the specified scale.	No	RO
ProcedureCache	Specifies the number of procedures which set the size of cache memory for parsed procedures.	10	RW/Startup
SimpleOptimizerRules	Instead of using full optimization rules, simplified one can be used by setting the value to "yes".	No	RW/Startup
SortArraySize	The size of the array that SQL uses when ordering the result set of a query. The units are "rows" — e.g. if you specify a value of 1000, then the server will create an array big enough to sort 1000 rows of data.	2000	RW/Startup
TimestampDisplaySize19	If this parameter is included in the `solid.ini` file and set to "Yes", it sets the precision (i.e. maximum number of digits) of data type timestamp to 19. In this case, the timestamp is presented as *yyyy-mm-dd hh:mm:ss*.	No	Startup

Table 45. SQL parameters (continued)

[SQL]	Description	Factory Value	Access Mode
TriggerCache	Specifies the number of triggers which set the size of cache memory that each user has for triggers.	20	RW/Startup
UpCaseQuotedIdentifiers	If set to yes, the SQL identifiers given in quotes are converted to upper case when reaching the solidDB server. If set to no, the upper/lower case distinction is preserved whereby uniqueness of names incorporates the case too.	Yes	RW/Startup

Srv section

Table 46. Srv parameters

[Srv]	Description	Factory Value	Access Mode
AbortTimeOut	Specifies the time in minutes after an idle transaction is aborted; negative or zero value means infinite.	120 Unit: 1 min	RW/Startup
AdaptiveRowsPerMessage	This parameter takes the average number of rows returned to the client as the rows per message value. The start value grows as more rows are fetched. If set to no, the RowsPerMessage parameter value is used. That is also the default value.	yes	RW/Startup
AllowConnect	If set to no, only connections from Remote Control or solidDB SQL Editor are allowed	yes	RW/Startup

Table 46. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode

At	The syntax is:
	`At = At_string`
	`At_string ::= timed_command`
	`[ ,timed_command ]`
	
	`timed_command ::=`
	`[ day ] HH:MM argument`
	`day ::= sun |`
	`mon |`
	`tue |`
	`wed |`
	`thu |`
	`fri |`
	`sat`
	
	If entered, allows you to specify a command to automate an administrative task, such as executing system commands, creating backups, checkpoints, and database status reports. For example:
	`At = 20:30 makecp,`
	`21:00 backup,`
	`sun 23:00 shutdown`
	
	If you specify a backup, the default backup directory is the one set with the BackupDirectory parameter in the General section.
	
	If the day is not given, the command is executed daily.
	
	There is no factory value for this parameter.
		(no factory value)	RW

ConnectionCheckInterval	When the ReadThreadMode parameter is set to 2 (default), the server doesn't detect a broken connection until it tries to write something back to the client. This parameter specifies the number of seconds between connection status checks in thread/client mode.	10 Unit: seconds	RW/Startup
ConnectTimeOut	Specifies the continuous idle time in minutes after a connection is dropped; negative or zero value means infinite.	480 Unit: 1 min	RW/Startup
DatabaseSizeReportInterval	When the database size exceeds the limit defined with this parameter, the system generates a report file. This parameter gives the delta after which the next report is printed. The minimum delta value is 1 MB. The report file name is `repdb<mb>MB.dbg`. This parameter is useful, for example when tracing unexpected database size growth. If you leave this parameter to its default value 0, no reports are generated. The minimum non-zero value for this parameter is 1 MB.	0 MB	RW/Startup

Table 46. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
DisableOutput	Disables generation of the `solmsg.out` and the `solerror.out` files. For details on these files, read "Viewing error messages and log files" on page 16. To disable file generation, this parameter must be included in the `solid.ini` file and set to yes. If this parameter is set to no or not included in the `solid.ini` file, the log files are generated.	no	RW/Startup
Echo	If set to yes, contents of `solmsg.out` file are displayed also at the server's command window.	no	RW/Startup
ExecRowsPerMessage	This parameter specifies how many result rows are sent (pre-fetched) to the client driver in response to the SQLExecute call with a SELECT statement. The result rows are subsequently returned to the application with the first SQLFetch calls issued by the application. The default value of 2 allows for pre-fetching of single-row results. If your SELECT statements usually return larger number of rows, setting this to an appropriate value can improve performance significantly. See also the RowsPerMessage configuration parameter.	2	RW/Startup
ForceThreadsToSystemScope	This parameter applies only to symmetric multi-process (SMP) Solaris operating systems, in which the default scope provided by the threads of the runtime library can be set to process scope, system scope, or light weight process (lwp) scope. (In Sun's terminology, "threads" are "lightweight processes".) A yes value may significantly improve the server's performance in a multi-CPU machine. (The actual performance improvement depends on how evenly the workload is already spread across your CPUs.) A no value usually provides slightly better performance in single-CPU systems. To fully understand how this parameter works, you must understand the threading facilities of Solaris. An explanation of the Solaris threading facilities is beyond the scope of this manual. However, it may be helpful to understand that when this parameter is set to yes, it forces lwp threads to be run in system scope, instead of process scope. A Yes setting allows Solaris to schedule solidDB threads on any available CPU. This reduces bottlenecks and enhances the parallelization of operations, including I/O.	Servers compiled for Solaris default to Yes. All other servers default to no.	RW/Startup

Table 46. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
HealthCheckEnabled	When the parameter is set to 'Yes', a periodical check is performed to detect a stalled server due to, for example, unexpected operating system stalls or software errors. The check uses a timeout-based server deadlock detection algorithm that checks certain critical low-level concurrent programming synchronization objects (mutexes). If a deadlock is detected, the server process terminates with an error and a message is printed to `solerror.out`. For example in High Availability (HotStandby) configurations, a failover can be enforced upon the detection of a server deadlock. **Note:** This parameter is not related to transaction-level deadlock detection mechanisms.	No	RW/Startup
HealthCheckInterval	This parameter sets the interval of the server deadlock check. Unit: seconds	60	RW
HealthCheckTimeout	This parameter sets the deadlock detection timeout time. The factory value is high enough to escape false errors. If faster detection is needed, set the parameter to a lower value. Unit: seconds.	60	RW
KeepAllOutFiles	If this parameter is set to yes, the solidDB message log (`solmsg.out`) and trace files are not overwritten with new contents. Instead, when a file limit is reached, a new file is created with an incremented file name number postfix. The starting value of the postfix is set by using parameters Srv.TraceBackupFileNum and Srv.SolmsgBackupFileNum.	No	RW/Startup
LocalStartTasks	Number of server's internal tasks (see footnote 1) that execute the local background statements that were started with command START AFTER COMMIT (without FOR EACH REPLICA). Valid values range from 1 - 100.	1	RW/Startup

Table 46. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode

MaxBgTaskInterval	This parameter (MAXimum BackGround TASK INTERVAL) tells the server the maximum length of time to wait before checking whether internal administrative tasks that are "sleeping" should be "awakened".
	
	The units are seconds.
	
	For example, if a connection has been broken or disconnected, this parameter specifies the maximum length of time that the server will wait before noticing that the connection is gone. This time is IN ADDITION TO whatever time is required for the underlying communication layer to detect that the connection is broken. For example, if you have a Connect Timeout of 100 seconds and a MaxBgTaskInterval of 50 seconds, then you may have to wait up to 150 seconds before a broken connection is detected and no longer counted as one of the connections.
	
	You may want to set or adjust this parameter if you get errors similar to the following:
	`Error 08004:`
	`[Solid][SOLID ODBC Driver]`
	
	`[SOLID]SOLID Server Error 14507:`
	`Maximum number of licensed user connections exceeded`
	
	This parameter only applies to the server's own internal administrative tasks. It does not affect the scheduling of user tasks.
	
	Attention: MaxBgTaskInterval applies to all server administration tasks, regardless of each task's priority. Even when a high priority task is running, the server will check the low-priority tasks at the specified intervals.
	
	Setting MaxBgTaskInterval to a small enough value may reduce performance and may reallocate some time from high-priority tasks to low-priority tasks. This is particularly likely to happen in "real-world" situations because the customers who use this parameter are most likely to be the customers with busy systems (that is, systems that were so busy they did not check low-priority connections often enough to notice that they had been disconnected). However, because the parameter only affects server administrative tasks, not user tasks, the effect is generally small.
		2 (seconds)	RW/Startup

Table 46. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
MaxConstraintLength	This parameter controls the maximum number of bytes that the server will search through in a string, for example in WHERE clauses such as: `WHERE LOCATE(sought_string, column1) > 0;` For example, if the value is 1024, ASCII character strings are searchable up to 1024 characters and Unicode character strings are searchable up to 512 characters (1024 bytes). This parameter applies to strings that have the following data types: CHAR(#) VARCHAR(#) It does not apply to strings that have the data type(s): LONG VARCHAR The minimum valid value is 254. If you specify a smaller number, the server will still search the first 254 bytes. Although you can use any value from 254 to 2G-1, practical values are generally in the range of a few kilobytes, like 1024, or 8192.	254 (254 bytes = 254 ASCII characters, or 127 Unicode characters)	RW
MaxOpenCursors	The maximum number of cursors that a database client can have simultaneously open.	1000	RW/Startup
MaxRPCDataLen	This allows users to specify the maximum string length of a single SQL statement sent to the server. This is particularly useful if you are sending CREATE PROCEDURE commands that are longer than 64K. The value should be between 64K (65536) and 1024K (1048576). If the value is less than 64K, the server will use a minimum of 64K.	512K (524288)	RW/Startup
MaxStartStatements	Maximum number of simultaneous "uncommitted" START AFTER COMMIT statements. Valid values range from 0 - 1000000.	10000	RW/Startup
MemoryReportLimit	This parameter defines the minimum size for memory allocations after which reporting to `solmsg.out` is done.	0 (no reporting)	RW/Startup
MemoryReportDelta	This parameter defines how much memory allocations must increase or decrease compared to the previous message before the new message is printed to `solmsg.out`.	20 MB	RW/Startup

Table 46. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
MemorySizeEventHysteresisPercentage	As the amount on memory used crosses different boundaries specified with, for example, the ImdbLowPercentage or the ProcessMemoryLimit parameter, system events are given. The event behavior expresses hysteresis in a way that the value triggering the `BELOW` event is somewhat lower than the specified value triggering the `ABOVE` event. The difference can be, for instance, 5%. As a result, the number of system events is not too large if the amount of memory alternates rapidly just above and below the specified boundaries. The MemorySizeEventHysteresisPercentage parameter is used to set the difference as a percentage value.	5	RW
MemorySizeReportInterval	When the memory size exceeds the limit defined with this parameter, the system generates a report file. This parameter defines the delta after which the next report is printed. The minimum delta value is 1 MB. The report file name is `repmem<mb>MB.dbg`. This is parameter is useful, for example when tracing unexpected memory growth in the server. If you leave this parameter to its default value 0, no reports are generated. The minimum non-zero value for this parameter is 1 MB.	0 MB	RW/Startup
MessageLogSize	The maximum size of the `solmsg.out` file in bytes.	1 MB Unit: 1 byte k=KB m=MB	RW/Startup
Name	Specifies the informal name of the server, equivalent to the `-n` command line option.		RW/Startup
NetBackupRootDir	Sets the root directory for the network backups in NetBackup Server. The path is relative to the working directory.	The working directory	RW
ODBCDefaultCharBinding	If set to UTF-8, ODBC-applications are allowed to store and retrieve UNICODE data in UTF-8 encoded format.	Raw	RW/Startup
PessimisticTableUseNFetch	Pessimistic table locks are used to prevent other sessions from adding, editing, or deleting any records or placing any record or table locks on a given table. Table locks block other record or table lock attempts, but do not block any reads of the locked table. If pessimistic tables are used, they force the RowsPerMessage value to 1 if the query locks any rows. You can enable the RowsPerMessage for pessimistic tables by enabling the PessimisticTableUseNFetch parameter. By default, it is disabled.	No	RW/Startup

Table 46. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
PrintMsgCode	Causes a unique 8-character message code to be inserted before each status and error message in the message log files (`solmsg.out` and `solerr.out`).	no	RW/Startup
ProcessMemoryCheckInterval	The process size limits are checked periodically. The check interval is set with the **ProcessMemoryCheckInterval** parameter. The interval is given in milliseconds. The minimum non-zero value is 1000 (ms). Only values 0 or 1000 or above 1000 (1 second) are allowed. If the given value is above 0 but below 1000, an error message is given. The factory value is 0, that is, process size checking is disabled. The **ProcessMemoryCheckInterval** also controls the **ProcessMemoryLimit** parameter; if the **ProcessMemoryCheckInterval** parameter value is 0, the **ProcessMemoryLimit** parameter is not effective, that is, there is no process memory limit. See also parameters **ProcessMemoryLowPercentage** and **ProcessMemoryWarningPercentage**.	0	RW
ProcessMemoryLimit	This parameter specifies the maximum amount of virtual memory that can be allocated to the in-memory database process. When this limit is exceeded, the server gives an error message and accepts admin commands only. The limit can be changed dynamically. The **ProcessMemoryLimit** parameter is controlled with the **ProcessMemoryCheckInterval** parameter; if the **ProcessMemoryCheckInterval** parameter value is 0, the **ProcessMemoryLimit** parameter is not effective	1G Unit: 1 byte, G=GB, M=MB, K=KB	RW

Table 46. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
ProcessMemoryLowPercentage	The **ProcessMemoryLowPercentage** parameter sets a warning limit for the total process size. The limit is expressed as percentage of the **ProcessMemoryLimit** parameter value. Prior to exceeding this limit, you have exceeded the warning limit defined by using the **ProcessMemoryWarningPercentage** parameter and received a warning. When the **ProcessMemoryLowPercentage** limit is exceeded, a system event is given. The **ProcessMemoryLowPercentage** parameter value is automatically checked for consistency. It must be higher than the **ProcessMemoryWarningPercentage** parameter value. See also parameters **ProcessMemoryLimit**, **ProcessMemoryCheckInterval** , and **ProcessMemoryWarningPercentage**.	90	RW
ProcessMemoryWarningPercentage	The **ProcessMemoryWarningPercentage** parameter sets the first warning limit for the total process size. The warning limit is expressed as percentage of the **ProcessMemoryLimit** parameter value. When the **ProcessMemoryWarningPercentage** limit is exceeded, a system event is given. The **ProcessMemoryWarningPercentage** parameter value is automatically checked for consistency. It must be lower than the **ProcessMemoryLowPercentage** parameter value. See also parameters **ProcessMemoryLimit**, **ProcessMemoryCheckInterval** , and **ProcessMemoryLowPercentage**.	80	RW

Table 46. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
ReadThreadMode	This parameter controls the number of threads that the server uses to service client requests. If the value is 0, the server uses the number of threads specified with the parameter Srv.Threads. If the value is 2, the server creates a separate thread for each client. Using more threads will generally improve performance, but also requires more memory. This parameter only controls the number of threads serving client requests. It does not affect the number of threads doing other work within the server. Some operating systems may limit the maximum number of threads allowed, and setting this parameter's value to 2 may cause the server to request more threads than the OS allows. If you try to exceed the number of threads allowed, you will get a message similar to the following: `"Failed to create thread 'dnet_clientthread'".` (msgcode 30146)	2	RW/Startup
RemoteStartTasks	Number of Replica server's internal tasks[1] inside the server that execute the remote background statements started at Master with command START AFTER COMMIT... FOR EACH REPLICA. Valid values range from 1 - 100.	1	RW/Startup
RowsPerMessage	Specifies the number of rows returned from the server in one network message when an SQLFetch call is executed (and there are no pre-fetched rows). See also the ExecRowsPerMessage configuration parameter.	100	RW/Startup
Silent	If set to yes, no output is generated to the server's command window. Only license information is displayed.	No	RW/Startup
SolmsgBackupFileNum	The starting value of the message log file (`solmsg.out`) name postfix appended to the file name if the Srv.KeepAllOutFiles parameter is set to yes. Valid values range from 0 to 999999	0	RW/Startup

Table 46. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
StandardDateTimeFormat	By default, solidDB uses the ISO/IEC/ANSI standard date representation, which is also the standard date literal format in SQL. The date is represented as shown in the timestamp example below: `2008-10-15 09:29:40` If you assign a "no" value to the StandardDateTimeFormat parameter, the message log files (`solmsg.out`) use a date presentation such as `15.10 09:29:40`. The `solerror.out` file uses another presentation, such as `Mon Oct 22 15:16:35 2007`.	yes	RW/Startup
StatementMemoryTraceLimit	This parameter switches on tracing for statements that have allocated memory over the defined value. These statements are put into the peak memory usage list. The peak memory list is printed to report file. Statements that use memory over the defined limit are also printed to the `solmsg.out` file.	0 MB	RW/Startup
Threads	If the Srv.ReadThreadMode parameter is set to 0, this parameter specifies the number of concurrent threads that the server uses to process user requests. The helper threads, such as I/O threads, are not included in the count. If the value of Srv.ReadThreadMode is other than 0, the value of this parameter is insignificant, as the server controls the number of threads automatically.	5	RW/Startup
TraceBackupFileNum	The starting value of the trace file name postfix appended to the file name if the Srv.KeepAllOutFiles parameter is set to yes. Valid values range from 0 to 999999	0	RW/Startup
TraceLogSize	This parameter allows you to limit the maximum size of the trace log file. The size is specified in bytes; for example, TraceLogSize=10000 limits the size of the trace log file to 10000 bytes. The trace log file is the file to which the server writes information when you turn on monitoring. (For information about turning on monitoring, see the description of ADMIN COMMAND 'monitor...' in Appendix B, "solidDB SQL Syntax", in *IBM solidDB SQL Guide*, and the `-m` command-line option in Appendix C, “solidDB command line options,” on page 175.) Monitoring uses the file named `soltrace.out` for output. Note that after reaching this maximum size, the server will: 1. delete any existing file named `soltrace.bak`; 2. rename the current `soltrace.out` file to `soltrace.bak`; and 3. start a new `soltrace.out` file.	1 megabyte Unit: 1 byte k=KB m=MB	RW/Startup

Table 46. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
TraceSecDecimals	Number of second decimals in trace outputs. Allowed values are from 0 to 3.	0	RW/Startup

[1] In this context, "task" means solidDB's internal task. Do not confuse this with "thread" or with the term "task" as it is used in some Real-Time Operating Systems such as Wind River Systems VxWorks. A task is just an operation that has to be executed, such as checkpoint, backup, or SQL statement. In this case, we can have 1 to N tasks that execute the background operations. More tasks mean that background tasks reserve more resources and are handled faster — and that other operations (for example, interactive ones) will get fewer resources and be handled more slowly.

Synchronizer section

Table 47. Synchronizer parameters

[Synchronizer]	Description	Factory Value	Access Mode
ConnectStrForMaster	This parameter indicates the connection string that the master must use to communicate with the replica. This information is read when the server is started, and sent to the master as part of each message from the replica to the master. For example, `ConnectStrForMaster= tcp replicahost 1316`	none	RW/Startup
MasterStatementCache	The size of the statement cache used during one propagation in Master. The statement cache is used to store prepared statements received by Master in one propagation from Replica.	10	RW/Startup

Table 47. Synchronizer parameters (continued)

[Synchronizer]	Description	Factory Value	Access Mode
RpcEventThresholdByteCount	This parameter controls how frequently the server posts events to indicate how many bytes have been sent or received in the current synchronization message. The units are measured in bytes; the smaller the value (that is, the smaller the number of bytes), the less frequently events are posted. Note that you cannot use suffixes such as "K" or "M" to indicate Kilobytes or Megabytes. The factory value is 0, which means that no events are posted. For more information, see the *IBM solidDB Advanced Replication User Guide*.	0	RW/Startup
RefreshIsolationLevel	With this parameter, you can select the transaction isolation level for refresh operations instead of using the `solid.ini` default value. The possible values are 1. READ COMMITTED 2. REPEATABLE READ	Defaults to SQL.IsolationLevel	RW/Startup
RefreshReadLevelRows	With this parameter, you can define the number of rows after the read level is released in the master if the used isolation level is READ COMMITTED. In other cases, the read level is kept for the full time of the refresh operation. The read level denotes a snapshot-consistent version of the data in the whole database. By releasing the read level, you avoid keeping too much data in main memory during the refresh operation.	1000	RW
Note:	The RemoteStartTasks parameter in the Srv section is also related to advanced replication.		RW/Startup

Table 47. Synchronizer parameters (continued)

[Synchronizer]	Description	Factory Value	Access Mode
ReplicaRefreshLoad	This parameter defines the amount of system processing capacity (as percentage) used to perform a refresh in Replica. By default, the full power is used. If some capacity is to be secured for local processing, in parallel with refresh, a lower value may be set.	100	RW

Appendix B. Client-side configuration parameters

The client-side configuration parameters are stored in the solid.ini configuration file and are read when the client starts.

Generally, the factory value settings offer the best performance and operability, but in some special cases modifying a parameter will improve performance. You can change the parameters by editing the configuration file solid.ini.

The parameter values set in the client side configuration file come to effect each time an application issues a call to the SqlConnect ODBC function. If the values are changed in the file during the program's run time, they affect the connections established thereafter.

Setting client-side parameters through the solid.ini configuration file

When the solidDB is started, it attempts to open the configuration file solid.ini. If the file does not exist, solidDB will use the factory values for the parameters. If the file exists, but a value for a particular parameter is not set in the solid.ini file, solidDB will use a factory value for that parameter. The factory values may depend on the operating system you are using.

By default, the client looks for the solid.ini file in the current working directory, which is normally the directory from which you started the client. When searching for the file, the solidDB uses the following precedence (from high to low):

- location specified by the SOLIDDIR environment variable (if this environment variable is set)
- current working directory

Rules for formatting the client-side solid.ini file

When you format the client-side solid.ini file, the same rules apply as for the server-side solid.ini file. For more information, refer to section "Rules for formatting the solid.ini file" on page 118.

Client-side solid.ini file

```
[Com]
;use this connect string of no data source given
Listen = tcp host1.acme.com 1315

[Client]
;at SQLConnect, timeout after this time (ms)
ConnectTimeout = 5000

;at any ODBC network request, timeout after this time (ms)
ClientReadTimeout = 10000

[DataSources]
Primary_Server = tcp irix1 1315, The Primary Server
Secondary_Server = tcp irix2 1315, The Secondary Server
```

Descriptions of client-side configuration parameters

There is one table below for each section of the `solid.ini` file. The sections (and tables) are:

- Com
- Data Sources
- Client

Communication section

Table 48. Communication parameters

[Com]	Description	Factory Value
ClientReadTimeout	This parameter defines the connection (or read) timeout in milliseconds. A network request fails if no response is received during the time specified. The value 0 sets the timeout to infinite. This value can be overridden with the connect string option -r and, further on, with the ODBC attribute SQL_ATTR_CONNECTION_TIMEOUT. Note: applies for the TCP protocol only.	60 000
Connect	The Connect parameter defines the default network name (connect string) for a client to connect to when it establishes a connection to a server. This value is used when the SQLConnect() call is issued with an empty data source name.	tcp localhost 1964
ConnectTimeout	The ConnectTimeout parameter defines the login timeout in milliseconds. This value can be overridden with the connect string option -c and, further on, with the ODBC attribute SQL_ATTR_LOGIN_TIMEOUT. Note: applies for the TCP protocol only.	OS-specific
ODBCHandleValidation	The ODBCHandleValidation parameter switches ODBC handle validation on/off. See also in section "ODBC Handle Validation" in *IBM solidDB Programmer Guide* for more information on the `SQL_ATTR_HANDLE_VALIDATION` ODBC attribute.	No
Trace	If this parameter is set to yes, trace information on network messages for the established network connection is written to a file specified with the TraceFile parameter. The factory value for the TraceFile parameter is `soltrace.out`.	no

Table 48. Communication parameters (continued)

[Com]	Description	Factory Value
TraceFile	If the Trace parameter is set to yes, trace information on network messages is written to a file specified with this TraceFile parameter.	`soltrace.out` (written to the current working directory of the server or client depending on which end the tracing is started)

Data sources

Table 49. Data source parameters

[Data Sources]	Description	Factory Value	Access Mode
logical name = network name, Description	These parameters can be used to give a logical name to a solidDB server in a `solid.ini` file of the client application. For details, read section "Logical Data Source Names" on page 108.		N/A

Client

Table 50. Client parameters

[Client]	Description	Factory Value
ExecRowsPerMessage	This parameter specifies how many result rows are sent (pre-fetched) to the client driver in response to the SQLExecute call with a SELECT statement. The result rows are subsequently returned to the application with the first SQLFetch calls issued by the application. The default value of 2 allows for pre-fetching of single-row results. If your SELECT statements usually return larger number of rows, setting this to an appropriate value can improve performance significantly. See also the RowsPerMessage configuration parameter.	decided by the server
NoAssertMessages	This parameter is relevant to the Windows platform only. If set to Yes, the Windows run-time error dialog is not shown.	No
ODBCCharBinding	If set to UTF-8, ODBC-applications are allowed to store and retrieve UNICODE data in UTF-8 encoded format.	Raw
RowsPerMessage	Specifies the number of rows returned from the server in one network message when an SQLFetch call is executed (and there are no pre-fetched rows). See also the ExecRowsPerMessage configuration parameter.	decided by the server

Table 50. Client parameters (continued)

[Client]	Description	Factory Value
StatementCache	Statement cache is an internal memory storing a few previously prepared SQL statements. With this parameter, you can set the number of cached statements per session.	6

Appendix C. solidDB command line options

Table 51. solidDB command line options

Option	Description	Examples
-c *dir*	Changes working directory.	`solid -c /data/solid`
-d *network_name*	Disables network name, that is, instructs the server not to listen for connections on this network name.	`solid tcp -d hobbes 1313`
-f	Starts the server in foreground.	
-h	Displays help.	
-m	Monitors users' messages and SQL statements.	
-n *name*	Sets the server name.	

-s install,*name*,*fullexepath* -c *workingdirectory*[,autostart]	The Microsoft Windows version of solidDB is by default an icon exe version. You can allow Windows to run solidDB as a service by using the option -s install.
	Note: After the service is installed, it must be started manually using the Windows Services dialog or command prompt.
	
	The [autostart] parameter sets the Startup Type of the service to *Automatic*, that is, solidDB will run automatically as a service when Windows is started. Note, however, that regardless of the [autostart] parameter, the service is not started automatically at the time of install. For the first time, the service has to be started manually in the Windows Services dialog or command prompt.
	
	When the server is running as a service, the server cannot interact with the display and cannot create a new database. The service version writes warning and error messages also to the Windows event log.
		`solid -s"install,SOLID, D:\SOLID\SOLID.EXE -cD:\SOLID"`
		
		`solid -s"install,SOLID, D:\SOLID\SOLID.EXE -cD:\SOLID,autostart"`

-s remove,*name*	Removes solidDB service.	`solid -s"remove,SOLID"`
-s start	Specifies that solidDB will start in a services mode when, for example, solidDB is created as a service using the Windows sc.exe utility. In the services mode, solidDB cannot interact with the display and cannot create a new database. **Note:** The - s start option is included automatically when using the -s install option.	`sc create SOLID binPath= "c:\soliddb\bin\solid.exe -cC:\soliddb -sstart"`

Table 51. solidDB command line options (continued)

Option	Description	Examples
-U *username*	See option -x execute or -x exit. If used without the -x option, specifies the username for the database being created.	
-P *password*	See option -x execute or -x exit. If used without the -x option, specifies the given password for the database being created.	
-C *catalog*	Specify the database catalog	
-E	Encrypt the database	
-S *password*	The database file encryption password.	
-x assert:s	Disables emergency exit dialog.	
-x autoconvert	Converts database format to the current format used by solidDB and starts the server process.	
-x convert	Converts database format to the current format used by solidDB and starts the server process.	
-x backupserver	See *IBM solidDB High Availability User Guide* for information.	
-x disableallmessageboxes	Hides all message windows	

-x decrypt -S *password*	Decrypts the database.	`solid -x decrypt -S dba`
		
		`solid -x decrypt -x keypwdfile:pwd.txt`

-x execute: *input file*	Prompts for the database administrator's user name and password, creates a new database, executes SQL statements from a file, and exits. The options -U and -P can be used to give the database the administrator's user name and password.
	
	The input file must be encoded with a 7-bit or 8-bit character set, such as ASCII or Latin-1.
		`solid.exe -x execute:init.sql`
		
		`solid.exe -x execute:init.sql -Udba -Pdba`

-x executeandnoexit: *input file*	Prompts for the database administrator's user name and password, creates a new database, executes SQL statements from a file, but does not exit.
	
	You can use this command with an existing database provided that you use options -U and -P to give the database the administrator's user name and password.
	
	The input file must be encoded with a 7-bit or 8-bit character set, such as ASCII or Latin-1.
		`solid.exe -x executeandnoexit: init.sql`
		
		`solid.exe -x executeandnoexit: init.sql -Udba -Pdba`

Table 51. solidDB command line options (continued)

Option	Description	Examples
-x exit	Prompts for the database administrator's user name and password, creates a new database, and exits. Options -U and -P can be used to give the database administrator's user name and password.	`solid.exe -x exit` `solid.exe -x exit -Udba -Pdba`
-x errormsgnostop	Does not wait for user actions on error dialogs.	
-x forcerecovery	Does a forced roll-forward recovery.	
-x hide	Hides the server icon.	
-x ignorecrashed	Ignores log files and reverts to checkpoint.	
-x ignoreerrors	Ignores index errors.	
-x infodbfreefactor	Informs about unused pages. See also:-x reorganize. The server exits after performing the task	
-x inifile: <file-name>	Substitutes an INI file.	
-x listen:<connect-string>	Sets a listening address.	
-x migratehsbg2	This command-line switch has two effects. It instructs the server to accept and convert the existing database (the same effect as the -x autoconvert parameter). Also, it enables the new Secondary to communicate with the old Primary by way of the old replication protocol. This parameter is needed only when upgrading a server that uses HotStandby.	
-x nologrecovery	With this command-line switch, you can ignore log files during recovery.	
-x pathprefix: <dir >	Uses files in the directory <dir>.	
-x pwdfile: *file name*	The password is read from the file name instead of command-line argument. This way the password can't be seen by running the UNIX command ps.	
-x keypwdfile: *file name*	The database encryption password is read from the file name instead of command-line argument. This way the password can't be seen by running the UNIX command ps.	
-x recreate_noconfirm	Creates a new empty database in place of the existing one.	
-x reorganize	Compacts the database by removing unused pages. The server exits after performing the task	

Table 51. solidDB command line options (continued)

Option	Description	Examples
-x testblocks	Tests database blocks and exits.	
-x testindex	Tests database index and exits.	
-x testintegrity	Performs a full database integrity test and exits.	
-x version	Displays the server version and exits.	
-?	Help = Usage.	
-h	Help = Usage.	

Appendix D. Error codes

This appendix lists error and message codes that can be generated by the server. This appendix lists the errors and messages according to the error class, following the order the error descriptions appear in the `ADMIN COMMAND 'errorcode all'` output.

Error classes

Table 52. solidDB error categories

Error class	Description
System	System errors are detected by the operating system and demand administrative actions. For the list of errors, see "solidDB system errors" on page 181.
Database or DBE (database engine)	The errors in these classes are detected by the solidDB and may demand administrative actions. Messages typically do not require administrative actions. For the list of errors and messages, see "solidDB database errors" on page 183 and "solidDB DBE (database engine) errors and messages" on page 239.
Table or TAB (table)	These errors and messages are caused by erroneous SQL statements detected by solidDB. Administrative actions are not needed. For the list of errors and messages, see "solidDB table errors" on page 192 and "solidDB TAB (table) messages" on page 247.
Communication, COM, Session, or RPC	The communication type errors are encountered by network problems, faulty configuration of the solidDB software, or ping facility errors. These errors in these classes usually demand administrative actions. Messages typically do not require administrative actions. For the list of errors and messages, see • "solidDB communication errors" on page 207 • "solidDB session errors" on page 206 • "solidDB COM (communication) messages" on page 237 • "solidDB RPC errors and messages" on page 219
Server	These errors are caused by erroneous administrative actions or client requests. They may demand administrative actions. For the list of errors, see "solidDB server errors" on page 210
Procedure	These errors are encountered when defining or executing a stored procedure. Administrative actions are not needed. For the list of errors, see "solidDB procedure errors" on page 216.
SA API	The SA API errors are return codes for the SA function SaSQLExecDirect. For more information, see "solidDB API errors" on page 219 and *SaSQLExecDirect* in the *IBM solidDB Programmer Guide*.
Sorter or XS	These errors are encountered when the external sorter algorithm is solving queries that require ordering rows. For the list of errors, see "solidDB sorter errors" on page 219 and "solidDB XS (external sorter) errors and messages" on page 246.

Table 52. solidDB error categories (continued)

Error class	Description
Synchronization or SNC	These errors may be encountered when creating or maintaining the solidDB environment. They occur when using certain solidDB statements that are solidDB SQL extensions. For the list of errors, see “solidDB synchronization errors” on page 221 and “solidDB SNC (synchronization) messages” on page 245.
HotStandby or HSB	The HotStandby errors occur when using the `ADMIN COMMAND 'HotStandby'` commands. For the list of errors, see “solidDB HotStandby errors” on page 234 and “solidDB HSB (HotStandby) errors and messages” on page 243.
SSA (solidDB SQL API)	These errors are caused by erroneous use of the solidDB SQL API (SSA). solidDB ODBC and JDBC drivers are implemented on this API. For the list of errors, see “solidDB SSA (SQL API) errors” on page 235
CP (checkpoint)	The CP messages provide information about the status or conditions of checkpoint operations. For the list of messages, see “solidDB CP (checkpoint) messages” on page 241.
BCKP (backup)	The BCKP messages provide information about the status or conditions of backup operations. For the list of messages, see “solidDB BCKP (backup) messages” on page 241.
AT (timed commands)	The AT messages provide information about the status or conditions of executing timed commands. For the list of messages, see “solidDB AT (timed commands) messages” on page 241.
LOG (logging)	The LOG messages provide information about the status or conditions of transaction logging. For the list of messages, see “solidDB LOG (logging) messages” on page 242.
INI (configuration file)	The INI messages provide information about the use of the `solid.ini` configuration file. For the list of messages, see “solidDB INI (configuration file) messages” on page 242.
FILE (file system)	The FILE messages provide information about file system operations, for example, for database and log files. For the list of messages, see “solidDB FIL (file system) messages” on page 246.
SQL errors	These errors are caused by erroneous SQL statements detected by the solidDB SQL Parser. Administrative actions are not needed. For the list of errors, see “solidDB SQL errors” on page 247
Executable errors	These errors are caused by the failure of the solidDB executable or a command line argument related error. They enable implementing intelligent error handling logic in system startup scripts. For the list of errors, see “solidDB executable errors” on page 254
solidDB Speed Loader (solload)	These errors are encountered when running the Speed Loader utility (solload) to load data from external files into the solidDB database. For the list of errors, see “solidDB Speed Loader (solload) errors” on page 255

In addition to the errors and messages described above, you might receive an internal error. In such a case, contact solidDB Technical Support at http://www.ibm.com/software/data/soliddb/support/.

solidDB system errors

Table 53. solidDB system errors

Code	Class	Type	Description
11000	System	Error	File open failure. The server is unable to open the database file. Reason for the failure can be: • The database file has been set to read-only. • You do not have rights to open the database file in write mode. • Another solidDB is using the database file. Correct the error and try again.
11001	System	Fatal Error	File write failure. The server is unable to write to the disk. The database files may have a read-only attribute set or you may not have rights to write to the disk. Add rights or unset read-only attribute and try again.
11002	System	Fatal Error	File write failed, disk full. The server failed to write to the disk, because the disk is full. Free disk space or move the database file to another disk. You can also split the database file to several disks using the FileSpec_[1-N] parameter in IndexFile section.
11003	System	Fatal Error	File write failed, configuration exceeded. Writing to the database file failed, because the maximum database file size set in FileSpec_[1-N] parameter is exceeded.
11004	System	Fatal Error	File read failure. An error occurred reading a file. This may indicate a disk error in your system.
11005	System	Fatal Error	File read beyond end of file. This error is given, if the file EOF is reached during the read operation.
11006	System	Fatal Error	File read failed, illegal file address. An error occurred reading a file. This may indicate a disk error in your system.
11007	System	Fatal Error	File lock failure. The server failed to lock the database file.
11008	System	Fatal Error	File unlock failure. The server failed to unlock a file.
11009	System	Fatal Error	File free block list corrupted. This error is given when reading data from disk to memory, but the memory space is already allocated for another purpose.
11010	System	Error	Too long file name. Filename specified in parameter FileSpec_[1-N] is too long. Change the name to a proper file name.

Table 53. solidDB system errors (continued)

Code	Class	Type	Description
11011	System	Error	Duplicate file name specification. Filename specified in parameter FileSpec_[1-N] is not unique. Change the name to a proper file name.
11012	System	Fatal Error	License information not found, exiting from solidDB Check the existence of your `solid.lic` file.
11013	System	Fatal Error	License information is corrupted. Your `solid.lic` file has been corrupted.
11014	System	Fatal Error	Database age limit of evaluation license expired.
11015	System	Fatal Error	Evaluation license expired.
11016	System	Fatal Error	License is for different CPU architecture.
11017	System	Fatal Error	License is for different OS environment.
11018	System	Fatal Error	License is for different version of this OS.
11019	System	Fatal Error	License is not valid for this server version.
11020	System	Fatal Error	License information is corrupted.
11021	System	Fatal Error	Problem with Your license, please contact IBM Corporation immediately.
11022	System	Error	Desktop license is only for local protocol communication, cannot use protocol for listening.
11023	System	Error	Internal binary stream error. This error is given if read or write fails when handling a binary stream object.
11024	System	Error	Desktop license is only for local communication, cannot use name for listening.
11025	System	Error	License file *filename* is not compatible with this server executable. The server has been started with an incompatible license file. You need to update your license file to match the server version.
11026	System	Error	Backup directory contains a file which could not be removed. Some file could not be removed from the backup directory. The backup directory may point to a wrong location.
11027	System	Error	No such parameter section *section*. Parameter was not found from the specified section in the `solid.ini` file.
11028	System	Error	No such parameter *section.name*. Parameter does not exist.

Table 53. solidDB system errors (continued)

Code	Class	Type	Description
11029	System	Error	Not allowed to set parameter value. User is not allowed to set the parameter value.
11030	System	Error	Cannot set values to multiple parameters. Only one parameter can be set at one time.
11031	System	Error	Illegal type for parameter. Parameter type is illegal.
11032	System	Error	Cannot set new value for parameter *section.name*. A new value cannot be set for the parameter.
11033	System	Error	Parameter is read-only.
11034	System	Error	File remove failure.
11035	System	Error	Value for parameter is smaller than minimum value.
11036	System	Error	Value for parameter is bigger than maximum value.
11037	System	Error	Value for parameter is invalid.
11038	System	Error	File specification exceeds the database address space.
11039	System	Error	File specification exceeds the database address space. This error is given if solidDB attempts to use a file, whose given size is larger that the size that solidDB can use.
11040	System	Error	Password file cannot be opened. This error is given if solidDB cannot find the database password file.
11041	System	Error	No password found in password file. This error is given if the database password is not in the password file.
11042	System	Error	Internal error: Empty diagnostic record. Contact technical support for more information.

solidDB database errors

Table 54. solidDB database errors

Code	Class	Type	Description
1004	Database	Warning	Database headers are inconsistent
1005	Database	Warning	Database is crashed
1012	Database	Warning	BLOB size overflow
1013	Database	Warning	BLOB size underflow
1019	Database	Return Code	Operation canceled

Table 54. solidDB database errors (continued)

Code	Class	Type	Description
1022	Database	Warning	The database you are using has been originally created with a different database block size setting than your current
10001	Database	Error	Key value is not found. Internal error: a key value cannot be found from the database index.
10002	Database	Error	Operation failed. This is an internal error indicating that the index of the table accessed is in inconsistent state. Try to drop and create the index again to recover from the error. You may also receive this error if you try to SET TRANSACTION READ ONLY when the transaction already contains some write operations.
10004	Database	Error	Redefinition. Unexpected failure occurred in the database engine. This error may also occur during recovery: either an index or a view has been redefined during recovery. The server is not able to do the recovery. Delete log files and start the server again.
10005	Database	Error	Unique constraint violation. You have violated a unique constraint. This happens when you have tried to insert or update a column which has a unique constraint and the value inserted or updated is not unique. This error message applies not only to user tables, but also to the system tables. For example, if you try to create a table that has the same name as an existing table, you may see this message. The same applies to other database object names, such as names of users, roles, triggers, etc.
10006	Database	Error	Concurrency conflict, two transactions updated or deleted the same row. Two separate transactions have modified a same row in the database simultaneously. This has resulted in a concurrency conflict.
10007	Database	Error	Transaction is not serializable. The transaction committed is not serializable.
10008	Database	Error	Snapshot does not exist.
10009	Database	Error	Snapshot is newest.
10010	Database	Fatal Error	No checkpoint in database. This error occurs when the server has crashed in the middle of creating a new database. Delete the database and log files and try to create the database again.
10011	Database	Fatal Error	Database headers are corrupted. The headers in the database are corrupted. This may be caused by a disk error or other system failure. Restore the database from the backup.
10012	Database	Fatal Error	Node split failed. This error is given if the node split of the in-memory database (B+ tree) fails.

Table 54. solidDB database errors (continued)

Code	Class	Type	Description
10013	Database	Error	Transaction is read-only. You tried to do one of the following: 1) Execute conflicting SET TRANSACTION statements, e.g. you executed SET TRANSACTION READ WRITE after you already SET TRANSACTION READ ONLY within the same transaction. 2) Write on a HotStandby database server that is in a Secondary state. 3) Write inside a transaction that is set read-only. Remove the write operation or unset the read-only mode in the transaction. If you see this message in the first transaction that you try to execute after connecting to a server, and if you haven't done anything to set the transaction or server to read-only mode, then try simply executing a COMMIT WORK statement and then re-executing the statement that caused the 10013 error.
10014	Database	Error	Resource is locked. This error occurs when you are trying to use a key value in an index which has been concurrently dropped.
10016	Database	Error	Log file is corrupted. One of the log files of the database is corrupted. You can not use these log files. Delete them and start the server again.
10017	Database	Error	Too long key value. The maximum length of the key value has been exceeded. The maximum value is one third of the size of the index leaf. If there are blobs (long varchars or long varbinaries) among the columns, the capacity requirements for a row can be reduced by storing the blob separately in the blob storage. However, when storing data in the blob storage, the first 254 bytes are also stored on the actual row. Therefore, with 8K block size, only 11 varchar columns with 254 characters of data is sufficient to exceed the key value limitation and cause this error message. You can try to: 1. Increase the [IndexFile] block size to increase the key value limit 2. Redesign your database to reduce space requirements. Design alternatives include: • Break columns with big VARCHAR strings to several rows in separate tables. Implement a view to represent the data accordingly. • Define columns with big VARCHAR strings to be concatenated inside one long VARCHAR to be processed as a blob. Implement a view to represent the data accordingly. 3. Define the table to be stored in the main memory. Since main memory storage uses a different algorithm, where the row size limitation is defined the by disk block size (minus overhead in the range of tens of bytes per row and few bytes per column), the limit is higher than with disk based tables. If the key value limit is exceeded in main memory tables, the error message is 16501.
10019	Database	Error	Backup is active You have tried to start a backup when a backup process is already in progress.
10020	Database	Error	Checkpoint creation is active. You have tried to start a checkpoint when a checkpoint creation is already in progress.

Table 54. solidDB database errors (continued)

Code	Class	Type	Description
10021	Database	Error	Failed to delete log file <log_file> (errno = <operating_system_error_code>. The deletion of a log file in making a backup has failed. Reasons for the failure can be: • The log file has already been deleted from the operating system. • The log file has a read-only attribute.
10023	Database	Fatal Error	Wrong log file, maybe the log file is from another database. The log file in the database directory is from another solidDB database. Copy the correct log files to the database directory.
10024	Database	Error	Illegal backup directory. The backup directory is either an empty string or a dot indicating that the backup will be created in the current directory.
10026	Database	Error	Transaction is timed out. An idle transaction has exceeded the maximum idle transaction time. The transaction has been aborted. The maximum value is set in parameter AbortTimeOut in SRV section. The default value is 120 minutes.
10027	Database	Error	No active search. This error is given during the UPDATE or DELETE operation if it is found that the active search identifying the data in the database to be updated or deleted does not exist.
10028	Database	Error	Referential integrity violation, foreign key values exist. You tried to delete a row that is referenced from a foreign key.
10029	Database	Error	Referential integrity violation, referenced column values do not exist. The definition of a foreign key does not uniquely identify a row in the referenced table.
10030	Database	Error	Backup directory '*directory name*' does not exist. Backup directory is not found. Check the name of the backup directory.
10031	Database	Error	Transaction detected a deadlock, transaction is rolled back. Deadlock detected. If necessary, begin transaction again.
10032	Database	Fatal Error	Wrong database block size specified. The block size of the database file differs from the block size given in the configuration file `solid.ini`.
10033	Database	Error	Primary key unique constraint violation. Your primary key definition is not unique.
10034	Database	Error	Sequence name *sequence* conflicts with an existing entity. Choose a unique name for a sequence. The specified name is already used.

Table 54. solidDB database errors (continued)

Code	Class	Type	Description
10035	Database	Error	Sequence does not exist. Check the name of the sequence.
10036	Database	Error	Data dictionary operation is active for accessed sequence. A create or drop operation is active for the accessed sequence. Finish the current transaction and then try again.
10037	Database	Error	Can not store sequence value, the target data type is illegal. The valid target data types are BIGINT, INTEGER, and BINARY.
10038	Database	Error	Illegal column value for descending index. Corrupted data found in descending index. Drop the index and create it again.
10039	Database	Error	INTERNAL: Assertion failure For more information, contact solidDB Technical Support at http://www.ibm.com/software/data/soliddb/support/.
10040	Database	Error	Log file write failure, probably the disk containing the log files is full. Shut down the server and reserve more disk space for log files.
10041	Database	Error	Database is read-only.
10042	Database	Error	Database index check failed, the database file is corrupted.
10043	Database	Error	Database free block list corrupted, same block twice in free list.
10044	Database	Error	Primary key can not contain blob attributes.
10045	Database	Error	This database is a HotStandby secondary server, the database is read only.
10046	Database	Error	Operation failed, data dictionary operation is active. Wait and try again.
10047	Database	Error	Replicated transaction is aborted.
10048	Database	Error	Replicated transaction contains schema changes, operation failed.
10049	Database	Error	Slave server not available any more, transaction aborted
10050	Database	Error	Replicated row contains BLOb columns that cannot be replicated.
10051	Database	Error	Log file is corrupted.
10052	Database	Fatal Error	Cannot convert an abnormally closed database. Please use the old solidDB database version to recover the database first.
10053	Database	Error	Table is read only.
10054	Database	Fatal Error	Opening the database file failed. Probably another solidDB process is already running in the same directory.

Table 54. solidDB database errors (continued)

Code	Class	Type	Description
10055	Database	Fatal Error	Too little cache memory has been specified for the solidDB process.
10056	Database	Fatal Error	Cannot open *database file. Error text (number)*. Most likely the solidDB process does not have correct access rights to the database file.
10057	Database	Fatal Error	The database is irrevocably corrupted. Revert to the latest backup.
10058	Database	Fatal Error	The internal database file format version (*number*) does not match with the solidDB version. Possible causes for this error include: • a version of solidDB that is too old is used with this database • the database has been corrupted
10059	Database	Fatal Error	The internal header version (*number*) does not match with the solidDB version. Possible causes for this error include: • a version of solidDB that is too old is used with this database • the database has been corrupted

10060	Database	Fatal Error	Cannot perform roll-forward recovery in read-only mode.
			Read-only mode can be specified in 3 ways. To restart solidDB in normal mode, verify that:
			• solidDB process is not started with command-line option `-x read only`
			• `solid.ini` does not contain the following parameter setting:
			`[General]`
			`ReadOnly=yes`
			• license file does not have read-only limitation

10061	Database	Fatal Error	Out of database cache memory blocks.
			solidDB process cannot continue because there is too little cache memory allocated for the solidDB process. Typical cause for this problem is a heavy load from several concurrent users. To allocate more cache memory, set the following `solid.ini` parameter to a higher value:
			`[IndexFile]`
			`CacheSize=cache_size_in_bytes`
			NOTE: Allocated cache memory size should not exceed the amount of physical memory.

10062	Database	Fatal Error	Failed to write to *log filename* at *offset*. Verify that the disk containing the log files is not full and is functioning properly. Also, log files should not be stored on shared disks over the network.
10063	Database	Fatal Error	Cannot create new log filename because such a file already exists in the log file directory. Probably your log file directory also contains logs from some other database. solidDB process cannot continue until invalid log files are removed from the log file directory. Remove *log filename* and all other log files with greater sequence numbers.

Table 54. solidDB database errors (continued)

Code	Class	Type	Description

10064	Database	Fatal Error	Illegal log file name template.
			Most likely, the log file name template specified in:
			`[Logging]`
			`FileNameTemplate=name`
			contains too few or too many sequence number digit positions. There should be at least 4 and at most 10 digit positions.

10065	Database	Fatal Error	Unknown log write mode. Please, recheck the configuration parameter.

10066	Database	Fatal Error	Cannot open *log filename*. Check the following log file name template in `solid.ini`:
			`[Logging]`
			`FileNameTemplate=name`
			and verify that:
			• it can be expanded into a valid file name in this environment
			• solidDB process has appropriate privileges to the log files directory.

10067	Database	Fatal Error	Cannot create database because old *log filename* exists in the log files directory. Possibly the database has been deleted without deleting the log files or there are log files from some other database in the log files directory of the database to be created.

10068	Database	Fatal Error	Roll-forward recovery cannot be performed because the configured log file *block size number* does not match with *block size number* of existing filename.
			To enable recovery, edit `solid.ini` to include parameter setting:
			`[Logging]`
			`BlockSize=blocksize in bytes`
			and restart the solidDB process. After successful recovery, you can change the log file block size by performing these steps:
			1. Shut down the solidDB process.
			2. Remove old log files.
			3. Edit new block size into `solid.ini`.
			4. Restart solidDB.

10069	Database	Fatal Error	Roll-forward recovery failed because *relation id number* was not found. Database has been irrevocably corrupted. Please restore the database from the last backup.
10070	Database	Fatal Error	Roll-forward failed because *relation id number* was not found. Database has been irrevocably corrupted. Please restore the database from the latest backup.
10071	Database	Fatal Error	Please restore the database from the latest backup.
10072	Database	Fatal Error	Database operation failed because of the file I/O problem.
10073	Database	Fatal Error	Database is inconsistent. Illegal index block type *size, address, routine, reachmode*. Please restore the database from the latest backup.
10074	Database	Fatal Error	Roll-forward recovery failed. Please revert to the latest backup.

Table 54. solidDB database errors (continued)

Code	Class	Type	Description

10075	Database	Fatal Error	The database you are trying to use has been originally created with different database block size settings than your current settings.
			
			Edit the `solid.ini` file to contain the following parameter setting:
			`[IndexFile]`
			`BlockSize=blocksize in bytes`

10076	Database	Fatal Error	Roll-forward recovery failed because *tablename* or *viewname* is redefined in the log filename. Possible causes for this error include: • another solidDB process is using the same log file directory • old log files are present in the log file directory solidDB process cannot use this corrupted log file to recover. In order to continue, you have the following alternatives: 1. Revert to the last backup 2. Revert to the last checkpoint 3. Revert to the last committed transaction within the last valid log file
10077	Database	Fatal Error	No base catalog given for database conversion (use -C *catalogname*) A database's base catalog must be provided when converting the database to a new format.
10078	Database	Error	User rolled back the transaction.
10079	Database	Error	Cannot remove filespec. File is already in use.
10080	Database	Error	HotStandby Secondary server can not execute operation received from Primary server. Meaning: A possible cause for this error is that the database did not originate from the Primary server using HotStandby copy or netcopy command.
10081	Database	Error	The database file is incomplete or corrupt. Meaning: If the file is on a hot standby secondary server, use the hotstandby copy or hotstandby netcopy command to send the file from the primary server again.
10082	Database	Error	Backup aborted.
10083	Database	Error	Failed to abort HSB transaction because commit is already sent to secondary.
10084	Database	Error	Table is not locked.
10085	Database	Error	Checkpointing is disabled.
10086	Database	Error	Deleted row not found. A key value being deleted cannot be found in the b-tree. This is an internal error.
10087	Database	Error	HotStandby not allowed for main memory tables.
10088	Database	Error	Specified lock timeout is too large.
10089	Database	Error	Operation failed, server is in HSB primary uncertain mode.
10090	Database	Error	Data dictionary operation in a newer transaction. This error is returned when a transaction tries to access a table whose schema has been altered by a later transaction. The recommended action is to retry the failing SQL command in a new transaction.
10091	Database	Error	Backup detected a log file with wrong block size, backup aborted.
10092	Database	Fatal Error	HotStandby cannot operate when logging is disabled.

Table 54. solidDB database errors (continued)

Code	Class	Type	Description
10093	Database	Fatal Error	HotStandby migration is not possible if Hotstandby is not configured.
10094	Database	Fatal Error	Only %d cache pages configured for M-table usage, at least %d needed.
10095	Database	Error	Cursor is closed after isolation change. The current cursor is closed, because its isolation level has been changed.
10096	Database	Fatal Error	Only <kilobytes> kilobytes configured for M-table checkpointing, at least <kilobytes>KB needed. Not enough memory has been configured for the M-table.
10098	Database	Error	Incrementing sequence *sequence_name* failed.
10099	Database	Fatal Error	Encryption password has not been given for encrypted database.
10100	Database	Fatal Error	Incorrect password has been given for encrypted database.
10101	Database	Fatal Error	Unknown encryption algorithm.
10104	Database	Fatal Error	Database is not created using solidDB Storage Engine for MySQL Prototype. Cannot open database.
10105	Database	Error	Cache size for hash table specified with <value> parameter is smaller than actual cache size.
10106	Database	Fatal Error	Too big cache memory has been specified for the SOLID process. Please edit the solid.ini file to change this parameter value not to exceed system limit and restart the SOLID process. This is a fatal error.
10107	Database	Error	Cursor is closed after logreader partition change.
10108	Database	Error	Search is aborted because of concurrent data dictionary operation on table.
16004	Database	Message	M-table operations now have enough memory for normal service.
16005	Database	Message	M-table operations now have enough memory for updates, inserts still disallowed.
16006	Database	Message	Memory for M-tables is now back below the warning level.
16501	Database	Error	New row value too large for M-table.
16502	Database	Error	BLOBs are not supported in M-tables.
16503	Database	Error	Serializable isolation level is not supported in M-tables.
16504	Database	Error	Memory for M-tables is running low, inserts to M-tables disallowed.
16505	Database	Error	Ran out of memory for M-tables, updates and inserts to M-tables disallowed.
16506	Database	Fatal Error	Too small configured **MME.ImdbMemoryLimit** to start server.
16507	Database	Error	Memory for M-tables is above the warning level.

solidDB table errors

Error code	Class	Type	Description
13001	Table	Error	Illegal character constant constant. An illegal character constant was found in the SQL statement.
13002	Table	Error	Type CHAR not allowed for arithmetic. You have entered a calculation having a character type constant. Character constants are not supported in arithmetic.
13003	Table	Error	Aggregate function not available for ordinary call. The aggregate function, such as SUM(), is called as an ordinary function. This is not allowed. For example, the following calls are illegal: SELECT * FROM TAB1 WHERE SUM(INT_COL) > 5; CALL SUM(1);
13004	Table	Error	Illegal aggregate function *parameter* parameter. An illegal parameter has been given to an aggregate function. Aggregate function parameters can only be column names or numbers.
13005	Table	Error	SUM and AVG not supported for CHAR type. Aggregate functions SUM and AVG are not supported for character type parameters.
13006	Table	Error	SUM or AVG not supported for DATE type. Aggregate functions SUM and AVG are not supported for date type parameters.
13007	Table	Error	Function *function* is not defined. The function you tried to use is not defined.
13008	Table	Error	Illegal parameter to ADD function.
13009	Table	Error	Division by zero. A division by zero has occurred.
13011	Table	Error	Table *table* does not exist. You have referenced a table which does not exist or you do not have REFERENCES privilege on the table.
13013	Table	Error	Table name *table* conflicts with an existing entity. Choose a unique name for a table. The specified name is already used.
13014	Table	Error	Index *index* does not exist. You have referenced an index which does not exist.
13015	Table	Error	Column *column* does not exist on table table. You have referenced a column in a table which does not exist.

Error code	Class	Type	Description
13018	Table	Error	Join table is not supported Joined tables are not supported in this version of solidDB.
13019	Table	Error	Transaction savepoints are not supported. Transaction savepoints are not supported in this version of solidDB.
13020	Table	Error	Default values are not supported. Default column values are not supported in this version of solidDB.
13022	Table	Error	Descending keys are not supported. Descending keys are not supported in this version of solidDB.
13023	Table	Error	Schema is not supported. Schema is not supported in this version of solidDB.
13025	Table	Error	Update through a cursor with no current row. You have tried to update using a cursor, but you do not have a current row in the cursor.
13026	Table	Error	Delete through a cursor with no current row You have tried to delete using a cursor, but you do not have a current row in the cursor.
13028	Table	Error	View *view_name* does not exist. You have referenced a view which does not exist.
13029	Table	Error	View name *view_name* conflicts with an existing entity. Choose a unique name for a view. The specified name is already used.
13030	Table	Error	No value specified for NOT NULL column column. You have not specified a value for a column which is defined NOT NULL.
13031	Table	Error	Data dictionary operation is active for accessed table or key. You can not access the table or key, because a data dictionary operation is currently active. Try again after the data dictionary operation has completed.
13032	Table	Error	Illegal type *type*. You have tried to create a table with a column having an illegal type.
13033	Table	Error	Illegal parameter *parameter* for type *type*. The type of the parameter you entered is illegal in this column.
13034	Table	Error	Illegal constant *constant*. You have entered an illegal constant.

Error code	Class	Type	Description
13035	Table	Error	Illegal INTEGER constant *constant*. You have entered an illegal integer type constant. Check the syntax of the statement and try again.
13036	Table	Error	Illegal DECIMAL constant *constant*. You have entered an illegal decimal type constant. Check the decimal number and try again.
13037	Table	Error	Illegal DOUBLE PREC constant *constant*. Typically, this is a general parse error. The SQL statement may contain a syntax error *before* the constant. As a last resort, the parser has attempted to parse a DOUBLE PREC constant, but has failed. This error also occurs if you entered an illegal double precision type constant. (More specifically, this error occurs when a space is placed between the asterisk and the closing parenthesis ("*)") in an optimizer hint.) In any of these cases, be sure to check the syntax of the statement and try again.
13038	Table	Error	Illegal REAL constant *constant*. You have entered an illegal real type constant. Check the real number and try again.
13039	Table	Error	Illegal assignment. You have tried to assign an illegal value for a column. For example, you may have tried to assign a value that was too large or was of the wrong data type.
13040	Table	Error	Aggregate *function* function is not defined. The aggregate function you tried to use is not supported.
13041	Table	Error	Type DATE not allowed for arithmetic. DATE type columns or constants are not allowed in arithmetic.
13042	Table	Error	Power arithmetic not allowed for NUMERIC and DECIMAL data type. Decimal and numeric data types do not support power arithmetic.
13043	Table	Error	Illegal date constant *constant*. A date constant is illegal. The correct form for date constants is: YYYY-MM-DD.
13046	Table	Error	Illegal user name *user*. User name entered is not legal. A legal user name is at least 2 and at most 31 characters in length. A user name may contain characters from A to Z, numbers from 0 to 9 and underscore character '_'.

Error code	Class	Type	Description
13047	Table	Error	No privileges for operation. You have no privileges for the attempted operation. To carry out this operation, you must be granted appropriate privileges. Alternatively, the operation can be performed by another user who already has the appropriate privileges. See the GRANT statement for more information. NOTE: If you are trying to drop a catalog that you previously created, and you get this error message, then your SYS_ADMIN_ROLE (i.e. DBA) privileges have been revoked. Only the creator of the database or users having SYS_ADMIN_ROLE (i.e. DBA) have privileges to create or drop a catalog. Even the creator of a catalog cannot drop that catalog if she loses SYS_ADMIN_ROLE privileges. (Creating a catalog, unlike creating most other objects (such as tables) does not make you the owner; instead, the ownership of all catalogs belongs to the DBA/ SYS_ADMIN_ROLE.)
13048	Table	Error	No grant option privilege for entity name. You have no privileges to grant privileges for the entity.
13049	Table	Error	Column privileges cannot be granted WITH GRANT OPTION Granting column privileges WITH GRANT OPTION is not supported in this version of solidDB.
13050	Table	Error	Too long constraint value. Maximum constraint length has been exceeded. Maximum constraint length is 255 characters.
13051	Table	Error	Illegal column name *column*. You have tried to create a table with an illegal column name.
13052	Table	Error	Illegal comparison operator operator for a pseudo column *column*. You have tried to use an illegal comparison operator for a pseudo column. Legal comparison operators for pseudo columns are: equality '=' and non-equality '<>'.
13053	Table	Error	Illegal data type for a pseudo column. You have tried to use an illegal data type for a pseudo column. Data type of pseudo columns is BINARY.
13054	Table	Error	Illegal pseudo column data, maybe data is not received using pseudo column. You have tried to compare pseudo column data with non-pseudo column data. Pseudo column data can only be compared with data received from a pseudo column.
13055	Table	Error	Update not allowed on pseudo column. Updates are not allowed on pseudo columns.
13056	Table	Error	Insert not allowed on pseudo column. Inserts are not allowed on pseudo columns.
13057	Table	Error	Index name *index* already exists. You have tried to create an index, but an index with the same name already exists. Use another name for the index.

Error code	Class	Type	Description
13058	Table	Error	Constraint checks were not satisfied on column column. Column has constraint checks which were not satisfied during an insert or update.
13059	Table	Error	Reserved system name *name*. You tried to use a name which is a reserved system name such as PUBLIC and SYS_ADMIN_ROLE.
13060	Table	Error	User name *user* not found. You tried to reference a user name which is not created.
13061	Table	Error	Role name *role* not found. You tried to reference a role name which is not created.
13062	Table	Error	Admin option is not supported. Admin option is not supported in this version of solidDB.
13063	Table	Error	Name *name* already exists. You tried to use a role or user which already exists. User names and role names must all be different, that is, you can not have a user named HOBBES and a role named HOBBES.
13064	Table	Error	Not a valid user name *user*. You tried to create an invalid user name. A valid user name has at least 2 characters and at most 31 characters. A user name may contain characters from A to Z, numbers from 0 to 9 and underscore character '_'.
13065	Table	Error	Not a valid role name *role*. You tried to create an invalid role name. A valid role name has at least 2 characters and at most 31 characters. A role name may contain characters from A to Z, numbers from 0 to 9 and underscore character '_'.
13066	Table	Error	User *user* not found in role *role*. You tried to revoke a role from a user and the user did not have that role.
13067	Table	Error	Too short password. You have entered a too short password. Password length must be at least 3 characters.
13068	Table	Error	Shutdown is in progress. You are unable to complete this operation, because server shutdown is in progress.
13070	Table	Error	Numerical overflow. A numerical overflow has occurred. Check the values and types of numerical variables.
13071	Table	Error	Numerical underflow. A numerical underflow has occurred. Check the values and types of numerical variables.

Error code	Class	Type	Description
13072	Table	Error	Numerical value out of range. A numerical value is out of range. Check the values and types of numerical variables.
13073	Table	Error	Math error. A mathematical error has occurred. Check the mathematics in the statement and try again.
13074	Table	Error	Illegal password. You have tried to enter an illegal password.
13075	Table	Error	Illegal role name *role*. You have tried to enter an illegal role name. A legal role name is at least 2 and at most 31 characters in length. A user role may contain characters from A to Z, numbers from 0 to 9 and underscore character '_'.
13077	Table	Error	Last column can not be dropped. You have tried to drop the final column in a table. This is not allowed; at least one column must remain in the table.
13078	Table	Error	Column already exist on table. You have tried to create a column which already exists in a table.
13079	Table	Error	Illegal search constraint. Check the search engine. There may be mismatch between data types.
13080	Table	Error	Incompatible types, can not modify column *column* fromtype *type* to type *type*. You have tried to modify column to a data type that isincompatible with the original definition, such as VARCHARand INTEGER
13081	Table	Error	Descending keys are not supported for binary columns. You can not define a descending key for a binary column.
13082	Table	Error	Function *function*: parameter * not supported. You can not use parameter star (*) with ODBC Scalar Functions.
13083	Table	Error	Function *function*: Too few parameters. The function expects more parameters. Check the function call.
13084	Table	Error	Function *function*: Too many parameters. The function expects fewer parameters. Check the function call.
13085	Table	Error	Function *function*: Run-time failure. An error was detected during the execution of the function. Check the parameters.

Error code	Class	Type	Description
13086	Table	Error	Function *function*: type mismatch in parameter parameter number. An erroneous type of parameter was detected in the given position of the function call. Check the function call.
13087	Table	Error	Function *function*: illegal value in parameter parameter number. An illegal value for a parameter detected in the given position of the function call. Check the function call.
13088	Table	Error	No primary key for table.
13090	Table	Error	Foreign key column *column* data type not compatible with referenced column data type. References specification error. Check that the column data type are compatible between referencing and referenced tables.
13091	Table	Error	Foreign key does not match to the primary key or unique constraint of the referenced table. References specification error. Check that the column data types are compatible between referencing and referenced tables and that the foreign key is unique for the referenced table.
13092	Table	Error	Event name *event* conflicts with an existing entity. Choose a unique name for an event. The specified name is already used.
13093	Table	Error	Event *event* does not exist. You referenced a nonexistent event. Check the name of the event.
13094	Table	Error	Duplicate column *column* in primary key definition. Duplicate columns are not allowed in a table-constraint-definition. Remove duplicate columns from the definition.
13095	Table	Error	Duplicate column *column* in unique constraint definition. Duplicate columns are not allowed in a table-constraint-definition. Remove duplicate columns from the definition.
13096	Table	Error	Duplicate column *column* in index definition. Duplicate columns are not allowed in CREATE INDEX statement. Remove duplicate columns.
13097	Table	Error	Primary key columns must be NOT NULL. Error in a *column_constraint_definition*. Define primary key columns NOT NULL. For example: CREATE TABLE DEPT (DEPTNO INTEGER NOT NULL, DNAME VARCHAR, PRIMARY KEY(DEPTNO));
13098	Table	Error	Unique constraint columns must be NOT NULL. Error in a *column_constraint_definition*. Define unique columns NOT NULL. For example: CREATE TABLE DEPT4 (DEPTNO INTEGER NOT NULL, DNAME VARCHAR, UNIQUE(DEPTNO));
13099	Table	Error	No REFERENCES privileges to referenced columns in table table. You do not have privileges to reference to the table.

Error code	Class	Type	Description
13100	Table	Error	Illegal table mode combination. You have defined an illegal combination of concurrency control settings. This message occurs, for example, if you have an in-memory table and you try to change it from pessimistic concurrency control (locking) to optimistic concurrency control by using the command ALTER TABLE <table_name> SET PESSIMISTIC. In-memory tables must always use pessimistic concurrency control.
13101	Table	Error	Only execute privileges can be used with procedures.
13102	Table	Error	Execute privileges can be used only with procedures.
13103	Table	Error	Illegal grant or revoke operation. This error occurs if you try to revoke privileges from yourself. This error occurs if the DBA tries to grant privileges to herself or himself (to the DBA).
13104	Table	Error	Sequence name *sequence* conflicts with an existing entity. Choose a unique name for a sequence. The specified name is already used.
13105	Table	Error	Sequence *sequence* does not exist. You referenced a nonexistent sequence. Check the name of sequence.
13106	Table	Error	Foreign key reference exists to table *table*.
13107	Table	Error	Illegal set operation. You tried to execute a non-existent set operation.
13108	Table	Error	Comparison between incompatible types *datatype* and *datatype*.
13109	Table	Error	There are schema objects for this user, drop failed
13110	Table	Error	NULL values given for NOT NULL column *column*.
13111	Table	Error	Ambiguous entity name *name*. This message occurs if the name of the specified database object (for example, a table name) does not exist in the schema that you are currently in, but more than one other schema contains an object with that name. If the database object that you want is in a different schema than the schema you are currently in, then change to the appropriate schema by using the SET SCHEMA command, or specify the desired object by using a more fully qualified object name, for example: sales_catalog.jan_wong_schema.table.1
13112	Table	Error	Foreign keys are not supported with main memory tables.
13113	Table	Error	Illegal arithmetic between types *datatype* and *datatype*.
13114	Table	Error	String operations are not allowed on values stored as BLOBs or CLOBs.
13115	Table	Error	Function *function_name*: Too long value (stored as CLOB) in parameter *parameter*. The parameter value was stored as CLOB and cannot be used with a function.

Error code	Class	Type	Description
13116	Table	Error	Column *column_name* specified more than once. Column was specified more than once in the GRANT or REVOKE statement.
13117	Table	Error	Wrong number of parameters Wrong number of parameters when converting subscription parameters to base publication parameter types.
13118	Table	Error	Column privileges are supported only for base tables. Column privileges are allowed only for base tables; they cannot be used, for example, for views.
13119	Table	Error	Types *column_type* and *column_type* are not union compatible. Column types are not union compatible. When a UNION operation is performed, two columns from two different tables are used to generate one column of output. The operation is successful as long as the two columns are of the same type or "compatible" types. Types are compatible if one type can reasonably be converted into the other type. For example, you can UNION a column of FLOAT with a column of INT because any integer value can also be represented as a corresponding float value (for example, 2 can be converted to 2.0). However, if you attempt a UNION operation on two incompatible types, such as FLOAT and DATE, you will receive 13119.
13120	Table	Error	Too long entity name *'entity_name'*. Entity name is too long, maximum entity name is 254 characters.
13121	Table	Error	Too many columns, maximum number of columns per table is *value*. Note that the maximum number of columns may be less if each column requires a large number of bytes.
13122	Table	Error	Operation is not supported for a table with sync history. Operation is not supported because the table has synchronization history defined.
13123	Table	Error	Table *'table_name'* is not empty. Some operations are allowed only for empty tables.
13124	Table	Error	User id *user_id* not found. Internal user id was not found; the user may have been dropped.
13125	Table	Error	Illegal LIKE pattern *'pattern'*. Illegal like pattern was given as a search constraint.
13126	Table	Error	Illegal type *datatype* for LIKE pattern. Only CHAR and WCHAR allowed for LIKE search constraints.
13127	Table	Error	Comparison failed because at least one of the values was too long. Comparison failed because at least one of the column values was stored as a BLOB or CLOB.

Error code	Class	Type	Description
13128	Table	Error	LIKE predicate failed because value is too long. LIKE predicate failed because the column value is stored as a CLOB.
13129	Table	Error	LIKE Predicate failed because pattern is too long. LIKE predicate failed because pattern value is stored as a CLOB.
13130	Table	Error	Illegal type *datatype* for LIKE ESCAPE character. Like ESCAPE character must be CHAR or WCHAR type.

13131	Table	Error	Too many nested triggers.
			
			Maximum number of nested triggers is reached. Triggers may be nested, for example, by activating other triggers from a trigger or causing recursive cycle when activating triggers. Default value for maximum allowed nested triggers is 16. It can be changed using a configuration parameter:
			`[SQL]`
			`MaxNestedTriggers=n`

13132	Table	Error	Too many nested procedures.
			
			Maximum number of nested procedures is reached. Procedures may be nested, for example, by activating other procedures from a procedure or causing a recursive cycle when activating procedures. Default value for maximum allowed nested procedures is 16. It can be changed using a configuration parameter:
			`[SQL]`
			`MaxNestedProcedures=n`

13133	Table	Error	Not a valid license for this product. The license file is for another solidDB product.
13134	Table	Error	Operation is allowed only for base tables. Given operation is available only for base tables.
13135	Table	Error	Internal error, arithmetic error in estimator For more information, contact solidDB Technical Support at http://www.ibm.com/software/data/soliddb/support/.
13136	Table	Error	Internal error, transaction is not active For more information, contact solidDB Technical Support at http://www.ibm.com/software/data/soliddb/support/.
13137	Table	Error	Illegal grant/revoke mode Grant or revoke mode is not allowed for given database objects.
13138	Table	Error	Index *index_name* given in index hint does not exist. Index name given in optimizer hint is not found for a table.
13139	Table	Error	Catalog *catalog_name* does not exist. Catalog name is not a valid catalog.

Error code	Class	Type	Description
13140	Table	Error	Catalog *catalog_name* already exists. Catalog name is an existing catalog.
13141	Table	Error	Schema *schema_name* does not exist. Schema name is not a valid schema.
13142	Table	Error	Schema *schema_name* already exists. Schema name is an existing schema.
13143	Table	Error	Schema *schema_name* is an existing user. Schema name specifies an existing user name.
13144	Table	Error	Commit and rollback are not allowed inside trigger. Commit or rollback are not supported inside trigger execution. This error is also given if a trigger calls a procedure that tries to execute commit or rollback command.
13145	Table	Error	Sync parameter not found. Parameter name given in command SET SYNC PARAMETER name NONE is not found.
13146	Table	Error	There are schema objects for this catalog, drop failed. Catalog contains schema object and cannot be dropped. Schema objects like tables and procedures need to be dropped before catalog can be dropped.
13147	Table	Error	Current catalog can not be dropped. The catalog that you want to drop must not be the current catalog. If you get this message, you should switch to another catalog, then re-execute the DROP CATALOG command.
13148	Table	Error	There are objects for this schema, drop failed.
13149	Table	Error	There are objects for this catalog, drop failed.
13150	Table	Error	Index can be created only into same catalog and schema as the base table.
13151	Table	Error	Cannot drop a column that is part of primary or unique key. Table definition contains a column that is part of a primary or unique key in an index.
13152	Table	Error	There are objects for this user, drop failed.
13153	Table	Error	Can not remove last administrator.
13154	Table	Error	Name cannot be an empty string.
13155	Table	Error	Column <column name> already exists on view <view name> The view definition contains the same column name twice.

Error code	Class	Type	Description
13156	Table	Error	Column attributes already exists on view.
13157	Table	Error	Current schema cannot be dropped.
13158	Table	Error	Current user cannot be dropped.
13160	Table	Error	Cannot alter table name because it is referenced in trigger(s). Altering the name of the table would prevent the trigger from working properly.
13161	Table	Error	An M-table is being updated with UPDATE ... WHERE CURRENT OF CURSOR and CURSOR is not declared FOR UPDATE. When you update an in-memory table (an "M-table") using the command UPDATE ... WHERE CURRENT OF CURSOR, you must have declared the cursor using the FOR UDPATE clause. This is required when the table is an in-memory table; it is strongly recommended, but not required, when the table is a disk-based table.
13162	Table	Error	A record in an M-table is being deleted with DELETE ... WHERE CURRENT OF CURSOR and CURSOR is not declared FOR UPDATE. When you delete a record from an in-memory table (an "M-table") using the command DELETE ... WHERE CURRENT OF CURSOR, you must have declared the cursor using the FOR UDPATE clause. This is required when the table is an in-memory table; it is strongly recommended, but not required, when the table is a disk-based table.
13163	Table	Error	Descending keys are not supported for bigint columns. If you try to create a DESCending index on a column of type BIGINT, you will get this message. Use an ASCending key instead.
13164	Table	Error	Transaction is active, operation failed.
13165	Table	Error	Can't fetch previous row from an M-table. This message can occur only when fetching rows from in in-memory table ("M-table") by using solidDB's low-level SA API.
13166	Table	Error	License does not allow accessing M-tables You will get this error message if you try to create an in-memory table and you do not have a license that allows you to do this. Generally, you need a license for solidDB disk-based engine to create in-memory tables.
13167	Table	Error	Only M-tables can be transient.
13168	Table	Error	Transient tables can not be set temporary.
13169	Table	Error	Temporary tables can not be set transient.
13170	Table	Error	Only M-tables can be temporary.
13171	Table	Error	Foreign key constraints between D- and M-tables are not supported.

Error code	Class	Type	Description
13172	Table	Error	A persistent table can not reference a transient table. For more details, see the discussion on persistent and transient tables under the CREATE TABLE command in the "Solid SQL Syntax" appendix in *solidDB SQL Guide*.
13173	Table	Error	A persistent table can not reference a temporary table. For more details, see the discussion on persistent and transient tables under the CREATE TABLE command in the "Solid SQL Syntax" appendix in *solidDB SQL Guide*.
13174	Table	Error	A transient table can not reference a temporary table. For more details, see the discussion on persistent and transient tables under the CREATE TABLE command in the "Solid SQL Syntax" appendix in *solidDB SQL Guide*.
13175	Table	Error	A reference between temporary and non-temporary table is not allowed.
13176	Table	Error	Cannot change STORE for a table with sync history.
13177	Table	Error	Cannot define UNIQUE constraint with duplicated or implied restriction.
13178	Table	Error	Constraint not found.
13179	Table	Error	Foreign key actions other than restrict are not supported.
13180	Table	Error	Constraint name already exists.
13181	Table	Error	Constraint check fails on existing data.
13182	Table	Error	Added column with NOT NULL must have a non-NULL default.
13183	Table	Error	Index is referenced by foreign key, it cannot be dropped.
13184	Table	Error	Primary key not found for table. Cannot define foreign key.
13185	Table	Error	Cannot set NOT NULL on column that already has NULL value.
13186	Table	Error	Cannot drop NOT NULL on column that is used as part of unique key.
13187	Table	Error	The cursor cannot continue accessing M-tables after the transaction has committed or aborted. The statement must be re-executed.
13188	Table	Error	Foreign key refers to itself.
13189	Table	Error	Positioning is not supported for M-tables.
13190	Table	Fatal Error	Definition in file is not valid.
13191	Table	Fatal Error	Parameter setting in file conflicts with the setting in database.
13192	Table	Fatal Error	Database is in read-only state

Error code	Class	Type	Description
13193	Table	Fatal Error	Foreign key creates update dependency loop. A foreign key creates a dependency between one or more tables in such a way that update to one row in one table might cause multiple updates to the same row in the same or another table. Such update might be ambiguous and the server does not allow creation of such dependencies. This restriction does not apply to cascaded deletes (when deletion of one row causes multiple deletions of another row), but it still applies when the deletion of one row causes multiple updates (SET NULL or SET DEFAULT) to another row.
13194	Table	Error	Can not drop a table that is part of a foreign key
13195	Table	Error	Update failed, READ COMMITTED isolation requires FOR UPDATE
13196	Table	Error	Delete failed, READ COMMITTED isolation requires FOR UPDATE
13197	Table	Error	M-tables are not supported
13198	Table	Error	Commit and rollback are not allowed inside function.
13199	Table	Error	Duplicate index definition This error is returned when a duplicate or redundant index is detected during index creation. For example, if you have created an index as follows: CREATE UNIQUE INDEX IND_1 ON T1(C1,C2,C3); Next, if you create this index: CREATE INDEX IND_2 ON T1(C2,C3,C1,C4); After this step, solidDB returns error 13199. In the example above, the second index is a superset of the unique first index. This implies that the second index (although it is not explicitly specified as unique) is also unique. In practice, the second index is useless. It only affects space consumption and update performance, not lookup performance.
13200	Table	Error	Update failed. Used isolation level requires FOR UPDATE.
13201	Table	Error	Delete failed. Used isolation level requires FOR UPDATE.
13202	Table	Error	Cluster connection does not support isolation levels higher than READ COMMITTED.
13203	Table	Error	License does not allow creating D-tables
13204	Table	Error	SET WRITE command makes sense only for TC connection
13205	Table	Error	Cannot change STORE for a table with foreign keys.
13400	Table	Error	Alter or drop table not allowed for propagated tables.
13401	Table	Error	Truncate table not allowed for propagated tables.
13402	Table	Error	Propagation information loading active.

Error code	Class	Type	Description
13403	Table	Error	Propagation information loading not active.
13404	Table	Error	Triggers not allowed for propagated tables.
13405	Table	Error	Cascading foreign keys not allowed for propagated tables.
13406	Table	Error	Primary key is required for propagated tables.
13407	Table	Error	Propagation schema data inconsistent: Table *name* not found.
13408	Table	Error	Logreader feature is disabled.
13409	Table	Error	Log overflow, catchup is not possible.
13410	Table	Error	Logreader partition not found .
13411	Table	Error	No active logreader query.
13412	Table	Error	Propagated tables allow only one row update when primary or unique key is changed.
13413	Table	Error	Blobs not supported for propagated tables.
13414	Table	Error	Given attribute value is incorrect for range partitioned table <value>.
13415	Table	Error	Range column <value> is not found from partitioned table <value>.
13416	Table	Error	Logreader partition already exists
13417	Table	Error	Table not found from logreader partition
13418	Table	Error	Table already exists in logreader partition
13501	Table	Warning	String data truncation in assignment from <value> to <value>
13502	Table	Warning	Numeric value right truncation in assignment from <value> to <value>

solidDB session errors

Table 55. solidDB session errors

Code	Class	Type	Description
20001	Session	Error	Illegal session class.
20002	Session	Error	Dynamic link library not found.
20003	Session	Error	Wrong dynamic link library version.
20004	Session	Error	Illegal address info.
20005	Session	Error	Listening address is in use.
20006	Session	Error	Server not found.
20007	Session	Error	Illegal control parameter.
20008	Session	Error	Illegal size parameter.

Table 55. solidDB session errors (continued)

Code	Class	Type	Description
20009	Session	Error	Write operation failed. This error is returned if the server or client is trying to write to an underlying communication channel (socket, named pipe, shared memory, etc.) that is broken.
20010	Session	Error	Read operation failed.
20011	Session	Error	Accept operation failed.
20012	Session	Error	Network not found.
20013	Session	Error	Out of network resources.
20023	Session	Error	Too many name resolver requests already in progress.
20024	Session	Error	Timeout while resolving host name.
20025	Session	Error	Timeout while connecting to a remote host.

solidDB communication errors

Table 56. solidDB communication errors

Code	Class	Type	Description
21100	Communication	Warning	Illegal value *value* for configuration *parameter* parameter, using default. An illegal value was given to the *parameter* parameter. The server will use a default value for this parameter.
21101	Communication	Warning	Invalid protocol definition *protocol* in configuration file. The protocol is defined illegally in the configuration file. Check the syntax of the definition.
21300	Communication	Error	Protocol *protocol* is not supported. Protocol is not supported.
21301	Communication	Error	Cannot load the dynamic link library *library* or one of its components. The server was unable to load the dynamic link library or a component needed by this library. Check the existence of necessary libraries and components.
21302	Communication	Error	Wrong version of dynamic link library *library*. The version of this library is wrong. Update this library to a newer version.
21303	Communication	Error	Network adapter card is missing or needed *protocol* software is not running. The network adapter card is missing or not functioning.
21304	Communication	Error	Out of protocol resources The network protocol is out of resources. Increase the protocols' resources in the operating system.

Table 56. solidDB communication errors (continued)

Code	Class	Type	Description
21305	Communication	Error	An empty or incomplete network name was specified. The network name specified is not legal. Check the network name.
21306	Communication	Error	Server *network name* not found, connection failed. The server was not found. 1) Check that the server is running. 2) Check that the network name is valid. 3) Check that the server is listening to the given network name.
21307	Communication	Error	Invalid connect info *network name*. The network name given as the connect info is not legal. Check the network name.
21308	Communication	Error	Connection is broken (*protocol read/write* operation failed with code *internal code*). The connection using the protocol is broken. Either a read or a write operation has failed with an internal error *internal code*.
21309	Communication	Error	Failed to accept a new client connection, out of *protocol* resources. The server was not able to establish a new client connection. The protocol is out of resources. Increase the protocol's resources in the operating system.
21310	Communication	Error	Failed to accept a new client connection, listening of *network name* interrupted. The server was not able to establish a new client connection. The listening has been interrupted.
21311	Communication	Error	Failed to start a selecting thread for *network name*. A thread selection has failed for *network name*.
21312	Communication	Error	Listening info *network name* already specified for this server. A network name has already been specified for this server. A server can not use a same network name more than once.
21313	Communication	Error	Already listening with the network name *network name*. You have tried to add a network name to a server when it is already listening with that network name. A server can not use a same network name more than once.
21314	Communication	Error	Cannot start listening, network name *network name* is used by another process. The server can not start listening with the given network name. Another process in this computer is using the same network name.
21315	Communication	Error	Cannot start listening, invalid listening info *network name*. The server can not start listening with the given listening info. The given network name is invalid. Check the syntax of the network name.
21316	Communication	Error	Cannot stop the listening of *network name*. There are clients connected. You can not stop listening of this network name. There are clients connected to this server using this network name.

Table 56. solidDB communication errors (continued)

Code	Class	Type	Description
21317	Communication	Error	Failed to save the listen information into the configuration file. The server failed to save this listening information to the configuration file. Check the file access rights and format of the configuration file.
21318	Communication	Error	Operation failed because of an unusual *protocol* return code *code*. Possible network error. Create connection again.
21319	Communication	Error	RPC request contained an illegal version number. Either the message was corrupted or there may be a mismatch between server and client versions.
21320	Communication	Error	Called RPC service is not supported in the server. There maybe a mismatch between server and client versions.
21321	Communication	Error	Protocol *protocol* is not valid, try using switch '-a' for specifying another adapter id instead of *switch*. This is returned if the NetBIOS LAN adapter id given in listen/connect string is not valid.
21322	Communication	Error	The host machine given in connect info '%s' was not found. This is returned in clients if the host machine name given in connect info is not valid.
21323	Communication	Error	Protocol *protocol* can not be used for listening in this environment. This message is displayed if the server end communication using specified protocol is not supported.
21324	Communication	Error	The process does not have the privilege to create a mailbox.
21325	Communication	Error	Only one listening name is supported in this server.
21326	Communication	Error	Failed to establish an internal *number* socket connection code *number*. solidDB uses one connect socket for internal use. Creation of this socket has failed; the local loopback may not be working correctly.
21327	Communication	Error	Too many name resolver requests already in progress.
21328	Communication	Error	Timeout while resolving host name.
21329	Communication	Error	Timeout while connecting to host.

solidDB server errors

Table 57. solidDB server errors

Code	Class	Type	Description
14003	Server	Return Code	ACTIVE ADMIN COMMANDs that may return this status in the result set of the command: • ADMIN COMMAND 'hotstandby status switch' • ADMIN COMMAND 'hotstandby status catchup' • ADMIN COMMAND 'hotstandby status copy' Meaning: The switch process, catchup process, copy or netcopy process is still active.
14007	Server	Return Code	CONNECTING ADMIN COMMANDs that may return this status in the result set of the command: • ADMIN COMMAND 'hotstandby status connect' Meaning: The Primary and Secondary servers are in the process of connecting.
14008	Server	Return Code	CATCHUP ADMIN COMMANDs that may return this status in the result set of the command: • ADMIN COMMAND 'hotstandby status connect' Meaning: The Primary server is connected to the Secondary server, but the transaction log is not yet fully copied. This message is returned only from the Primary server.
14009	Server	Return Code	No server switch occurred before. ADMIN COMMANDs that may return this status in the result set of the command: • ADMIN COMMAND 'hotstandby status switch' Meaning: The switch process has never happened between the servers.
14501	Server	Error	Operation failed. This error occurs when a timed command fails. Check the arguments of timed commands. This error number is also used for certain HotStandby errors. See *IBM solidDB High Availability User Guide* for details.
14502	Server	Error	RPC parameter is invalid. A network error has occurred.
14503	Server	Error	Communication error. A communication error has occurred.
14504	Server	Error	Duplicate cursor name cursor. You have tried to declare a cursor with a cursor name which is already in use. Use another name.
14505	Server	Error	Connect failed, illegal user name or password. You have entered either a user name or a password that is not valid.
14506	Server	Error	The server is closed, no new connections allowed. You have tried to connect to a closed server. Connecting was aborted.

Table 57. solidDB server errors (continued)

Code	Class	Type	Description
14507	Server	Error	Maximum number of licensed user connections exceeded. You have tried to connect to a server which has all licenses currently in use. Connecting was aborted.
14508	Server	Error	The operation has timed out. You have launched an operation that has been aborted.
14509	Server	Error	Version mismatch. A version mismatch has occurred. The client and server are different versions. Use same versions in the client and the server.
14510	Server	Error	Communication write operation failed. A write operation failed. This indicates a network problem. Check your network settings.
14511	Server	Error	Communication read operation failed. A read operation failed. This indicates a network problem. Check your network settings.
14512	Server	Error	There are users logged to the server. You can not shutdown the server now. There are users connected to the server.
14513	Server	Error	Backup process is active. You cannot shut down the server now. The backup process is active
14514	Server	Error	Checkpoint creation is active. You cannot shut down the server now. The checkpoint creation is active.
14515	Server	Error	Invalid user id. You tried to drop a user, but the user id is not logged in to the server.
14516	Server	Error	Invalid user name. You tried to drop a user, but the user name is not logged in to the server.
14517	Server	Error	Someone has updated the at commands at the same time, changes not saved. You tried to update timed commands at the same time another user was doing the same. Your changes will not be saved.
14518	Server	Error	Connection to the server is broken, connection lost. Possible network error. Reconnect to the server.
14519	Server	Error	The user was thrown out from the server, connection lost. Possible network error.
14520	Server	Error	Server is HotStandby secondary server, no connections are allowed.
14521	Server	Error	Failed to create a new thread for the client.

Table 57. solidDB server errors (continued)

Code	Class	Type	Description
14522	Server	Error	HotStandby copy directory not specified. Meaning: No copy directory is specified. ADMIN COMMANDs that may return this status in the result set of the command: • ADMIN COMMAND 'hotstandby copy' To solve this problem, either specify the directory as part of the command, for example: `ADMIN COMMAND 'hotstandby copy \Secondary\dbfiles\'` or else set the **CopyDirectory** parameter in the `solid.ini` configuration file.
14523	Server	Error	Switch process is already active. Meaning: The switch process is already active in the HotStandby server. If you only need to complete the current switch, then wait. If you are trying to switch a second time (that is, switch back to the original configuration), then you must wait for the first switch to complete before you can start the second switch. ADMIN COMMANDs that may return this status in the result set of the command: • ADMIN COMMAND 'hotstandby switch primary' • ADMIN COMMAND 'hotstandby switch secondary' • ADMIN COMMAND 'hotstandby status switch'
14524	Server	Error	HotStandby databases have a different base database, database time stamps are different. Meaning: Databases are from a different seed database. You must synchronize databases. You may need to perform netcopy of the Primary's database to the Secondary. ADMIN COMMANDs that may return this status in the result set of the command: • ADMIN COMMAND 'hotstandby connect' • ADMIN COMMAND 'hotstandby status switch'
14525	Server	Error	HotStandby databases are not properly synchronized. Meaning: Databases are not properly synchronized. You must synchronize the databases. You may need to start one of the database servers (the one that you intend to become the Secondary) with the command line parameter -x backupserver and then netcopy the Primary's database to the Secondary. ADMIN COMMANDs that may return this status in the result set of the command: • ADMIN COMMAND 'hotstandby connect' • ADMIN COMMAND 'hotstandby status switch'
14526	Server	Error	Invalid argument. Meaning: An argument used in the HotStandby ADMIN COMMAND is unknown or invalid. All HotStandby commands can return this error in the result set of the ADMIN COMMAND. Note: In the following HotStandby commands, the invalid argument error is a syntax error when the specified Primary or Secondary server can not apply to the switch: • ADMIN COMMAND 'hotstandby switch primary' • ADMIN COMMAND 'hotstandby switch secondary'

Table 57. solidDB server errors (continued)

Code	Class	Type	Description
14527	Server	Error	This is a non-HotStandby server. Meaning: The command was executed on a server that is not configured for HotStandby. ADMIN COMMANDs that may return this status in the result set of the command: • ADMIN COMMAND 'hotstandby connect' • ADMIN COMMAND 'hotstandby status switch' • ADMIN COMMAND 'hotstandby switch primary' • ADMIN COMMAND 'hotstandby switch secondary' • ADMIN COMMAND 'hotstandby state'
14528	Server	Error	Both HotStandby databases are primary databases. Meaning: Both databases are Primary. This is a fatal error because there may be conflicting changes. Both databases are automatically dropped to Secondary state by the system. You must decide which database is the real Primary database and then synchronize the databases. ADMIN COMMANDs that may return this status in the result set of the command: • ADMIN COMMAND 'hotstandby connect' • ADMIN COMMAND 'hotstandby status switch'
14529	Server	Error	The operation timed out.
14530	Server	Error	The connected client does not support UNICODE data types. Connected client is an old version client that does not support UNICODE data types. UNICODE data type columns cannot be used with old clients.

14531	Server	Error	Too many open cursor, max limit is *value*.
			
			There are too many open cursors for one client; maximum number of open cursors for one connection is 1000. The value can be changed using a configuration value:
			`[Srv]`
			`MaxOpenCursors=n`

14532	Server	Error	Internal error: cursor synchronization between client and server failed. Contact technical support for more information.
14533	Server	Error	Operation cancelled Operation was cancelled because client application called ODBC or JDBC cancel function.
14534	Server	Error	Only administrative statements are allowed. Only administrative statements are allowed for the connection.
14535	Server	Error	Server is already a primary server. Meaning: The server you are trying to switch to Primary is already in one of the PRIMARY states. ADMIN COMMANDs that may return this status in the result set of the command: • ADMIN COMMAND 'hotstandby switch primary'

Table 57. solidDB server errors (continued)

Code	Class	Type	Description
14536	Server	Error	Server is already a secondary server. Meaning: The server you are trying to switch to Secondary is already in one of the SECONDARY states. ADMIN COMMANDs that may return this status in the result set of the command: • ADMIN COMMAND 'hotstandby switch secondary'
14537	Server	Error	HotStandby connection is broken. Meaning: This command is returned from both the Primary and Secondary server. ADMIN COMMANDs that may return this status in the result set of the command: • ADMIN COMMAND 'hotstandby status connect' One possible cause of this problem is an incorrect Connect string in the Secondary's `solid.ini` file. If the netcopy operation succeeds but the connect command fails, check the Connect string. (Netcopy does not require the Secondary to open a separate connection to the Primary, and thus may succeed even if the Connect string on the Secondary is wrong.)
14538	Server	Error	Server is not HotStandby primary server. Meaning: To issue this command, the server must be a HotStandby Primary server. ADMIN COMMANDs that may return this status in the result set of the command: • ADMIN COMMAND 'hotstandby copy copy_directory' • ADMIN COMMAND 'hotstandby netcopy' • ADMIN COMMAND 'hotstandby connect' • ADMIN COMMAND 'hotstandby set primary alone' • ADMIN COMMAND 'hotstandby set standalone'
14539	Server	Error	Operation Refused. This error code is given when one of the following situations occurs: • The user issued a netcopy command to a Primary server, but the server that should be Secondary is not actually in a Secondary state, or is not in "netcopy listening mode". (Both the Primary and the "Secondary" server are probably in PRIMARY ALONE state.) To solve the problem, restart the "Secondary" with the -x backupserver command-line option, then try again to issue the netcopy command to the Primary. **Attention:** If both servers were in PRIMARY ALONE state, and if both servers executed transactions while those servers were in PRIMARY ALONE state, then they probably each have data that the other one does not. This is a serious error, and doing a netcopy to put them back in sync would result in writing over some transactions that have already been committed in the "Secondary" server. • This message can be generated when you use a callback function and the callback function refuses to shut down or accept a backup or netcopy command. When you use linked library access, you can provide "callback" functions by using the SSCSetNotifier function. Your callback functions will be notified when the server has been commanded to shut down or to do a netcopy operation. If for some reason your application doesn't want the command to be followed, then your callback can return a value that cancels the command. In this situation, you will see error 14539. To solve the problem, wait until the client code finishes the operation that it does not want to interrupt, then retry the command (for example, the shutdown or netcopy).
14540	Server	Error	Server is already a non-HotStandby server.
14541	Server	Error	HotStandby configuration in `solid.ini` conflicts with ADMIN COMMAND 'HSB SET STANDALONE'.
14542	Server	Error	Server in backupserver mode. Operation refused.
14543	Server	Error	Invalid command. The database is a HotStandby database but, HotStandby section not found in `solid.ini` configuration file.
14544	Server	Error	Operation failed. This command is not supported on diskless server.

Table 57. solidDB server errors (continued)

Code	Class	Type	Description
14545	Server	Error	Primary can only be set to primary alone when its role is primary broken.
14546	Server	Error	Switch failed. The server or the remote server cannot switch from primary alone to secondary server. Catchup should be done first before switch. Meaning: This command is returned when a state switch to SECONDARY is executed from a local or remote Primary server that is in the PRIMARY ALONE state and it is detected that the Primary and Secondary server are not in sync. You must connect the Primary server to the Secondary server and wait for the catchup process to complete before switching the Secondary to the Primary. HotStandby commands that return this error: • ADMIN COMMAND 'hotstandby switch secondary'
14547	Server	Error	The value for the -R option (Read Timeout) was missing or invalid.
14548	Server	Error	Switch failed. The server in Standalone cannot be switched to a secondary. Meaning: This command is returned when a state switch to SECONDARY is executed from a local or remote Primary server that is in the STANDALONE state and it is detected that the Primary and Secondary server are not in sync. You must connect the Primary server to the Secondary server and wait for the catchup to complete before switching the Secondary to the Primary. HotStandby commands that return this error: • ADMIN COMMAND 'hotstandby switch secondary'
14549	Server	Error	HotStandby transaction is active. Meaning: If the HotStandby connection is broken, Primary server must be set to alone mode or switched to secondary mode before shutdown.
14550	Server	Error	Hotstandby connect parameter can be changed only when the primary is not connected to secondary.
14551	Server	Error	Maximum number of START AFTER COMMIT statements reached.
14552	Server	Error	Server is in backup server mode, no connections are allowed. Error 14552 is returned when a client attempts to establish a connection to a solidDB server which is in a backup server mode (also called *netcopy listening mode*). The backup server mode is a special server mode where the solidDB instance has been started with the command line option `-xbackupserver`. This mode indicates that the solidDB instance is a Secondary server that is either waiting for or in the process of receiving the database file from the Primary server due to a **netcopy** command issued at the Primary server.
14553	Server	Error	Backup process is not active This error is given if ADMIN COMMAND 'abort backup' is issued and no backup is active.
14554	Server	Error	The server does not support the required Transparent Failover level. Reserved for future. This error will be reported when the server does not implement the Transparent Failover (TF) level requested by the application. Currently, there is only one level.
14555	Server	Error	Netbackup: Conflicting usage of backup directory %s.
14556	Server	Error	Netbackup: No server connection string specified.
14557	Server	Error	Netbackup: A server configured for HotStandby cannot act as a netbackup server.
14600	Server	Error	Command is ambiguous in cluster session.
14706	Server	Error	Invalid read thread mode for HotStandby, only mode 2 is supported.

solidDB procedure errors

Code	Class	Type	Description
23001	Procedure	Error	Undefined symbol *symbol*
23002	Procedure	Error	Undefined cursor *cursor*. You have used a cursor that has not been defined in a procedure definition.
23003	Procedure	Error	Illegal SQL operation *operation*.
23004	Procedure	Error	Syntax error: parse error, line *line number*. Check the syntax of your procedure.
23005	Procedure	Error	Procedure *procedure* not found.
23006	Procedure	Error	Wrong number of parameters for procedure procedure.
23007	Procedure	Error	Procedure name *value* conflicts with an existing entity. Choose a unique name for a procedure. The specified name is already used.
23010	Procedure	Error	Incompatible event *event* parameter type, line *line number*.
23011	Procedure	Error	Wrong number of parameter for event *event*, line *line number*.
23012	Procedure	Error	Duplicate wait for event *event*, line *line number*.
23013	Procedure	Error	Undefined sequence *sequence*.
23014	Procedure	Error	Duplicate sequence name *sequence*.
23015	Procedure	Error	Sequence *sequence* not found.
23016	Procedure	Error	Incompatible variable type in call to sequence *sequence*, line *line number*.
23017	Procedure	Error	Duplicate symbol *symbol*. You have duplicate definitions for a symbol.
23018	Procedure	Error	Procedure owner *owner* not found.
23019	Procedure	Error	Duplicate cursor name *'cursor'*
23020	Procedure	Error	Illegal option *option* for WHENEVER SQLERROR ... statement.
23021	Procedure	Error	RETURN ROW not allowed in procedure with no return type, line *line number*.
23022	Procedure	Error	SQL String variable variable must be of character data type, line *line number*.
23023	Procedure	Error	Call syntax error: *syntax*, line *line number*.
23024	Procedure	Error	Trigger *trigger_name* not found. Trigger name not found.

Code	Class	Type	Description
23025	Procedure	Error	Trigger name *trigger_name* conflicts with an existing entity. Trigger name conflicts with some other database object. Triggers share the same name space, as for example, in table and procedures.
23026	Procedure	Error	Variable *variable* is of character type, line line number. A CHAR or WCHAR variable is required for the operations like RETURN SQLERROR variable.
23027	Procedure	Error	Duplicate reference to column *column_name* in trigger definition. One column can be referenced only once in the trigger definition.
23028	Procedure	Error	Commit and rollback are not allowed in triggers. Trigger body may not contain commit or rollback statements.
23029	Procedure	Error	Commit and rollback are not allowed in functions.
23030	Procedure	Error	Function *function_name* not found
23501	Procedure	Error	Cursor *cursor* is not open.
23502	Procedure	Error	Illegal number of columns in EXECUTE ... *procedure* in cursor *cursor*. You will see this message if the number of columns that you selected does not match the number of variables in the INTO clause.
23503	Procedure	Error	Previous SQL operation *operation* failed in cursor cursor.
23504	Procedure	Error	Cursor *cursor* is not executed.
23505	Procedure	Error	Cursor *cursor* is not a SELECT statement.
23506	Procedure	Error	End of table in cursor *cursor*.
23508	Procedure	Error	Illegal assignment, line *line number*.
23509	Procedure	Error	In *procedure* line *line number* Stmt *statement* was not in error state in RETURN SQLERROR OF ...
23510	Procedure	Error	In *procedure* line *line number* Transaction cannot be set read only, because it has written already.
23511	Procedure	Error	In *procedure* line *line number* USING part is missing for dynamic parameters for *procedure*.
23512	Procedure	Error	In *procedure* line *line number* USING list is too short for *procedure*.
23513	Procedure	Error	In *procedure* line *line number* Comparison between incompatible types *data type* and *data type*.
23514	Procedure	Error	In *procedure* line *line number* type *data type* is illegal for logical expression.

Code	Class	Type	Description

23515	Procedure	Error	In *procedure* line *line number* assignment of *parameter* parameter in list *list* failed.
			
			One possible cause of this error is trying to bind a parameter in a prepared statement that has a clause like "...? IS NULL...". To work around this problem, we recommend that you cast the placeholder (the question mark) to the appropriate data type. For example, if you are binding a parameter of type TIMESTAMP, then replace
			`WHEN ? IS NULL`
			
			with
			`WHEN CAST(? AS TIMESTAMP) IS NULL`

23516	Procedure	Error	In CALL *procedure*, assignment of parameter *parameter* failed.
23517	Procedure	Error	Internal error: illegal operation code in procedure. Contact technical support for more information.
23518	Procedure	Error	User error: *error_text* User generated error in a procedure or trigger. User can generate this error by using a statement RETURN SQLERROR *string* or RETURN SQLERROR *variable*. Variable must be of CHAR or WCHAR type.
23519	Procedure	Error	Fetch previous is not supported for procedures. Fetch previous row does not work for result sets returned by a procedure.
23520	Procedure	Error	Invalid link name given in remote procedure call.
23521	Procedure	Error	Link name not given in remote procedure call.
23522	Procedure	Error	Dynamic parameters not allowed with remote procedure call.
23523	Procedure	Error	Default node not defined.
23524	Procedure	Error	Could not load application.
23525	Procedure	Error	Function not found from the DLL.
23526	Procedure	Error	In CALL <procedure_name> assignment of default value of parameter <parameter_number> failed. This error message occurs if you call a procedure with too few parameters and you have not specified default values for the missing parameters.
23527	Procedure	Error	In CALL <procedure_name> parameter <parameter_number> assigned twice. This occurs if you specify the same parameter more than once.
23528	Procedure	Error	Application is already running.
23529	Procedure	Error	Application is not running.

solidDB API errors

Table 58. solidDB SA API errors

Code	Class	Type	Description
15001	API	Error	Syntax error: <error>, <line>.
15002	API	Error	Illegal column name <name>.
15003	API	Error	Too many parameters for string constraints.
15004	API	Error	Too few parameters for string constraints.

solidDB sorter errors

Table 59. solidDB sorter errors

Code	Class	Type	Description
24001	Sorter	Error	Sort failed due to insufficient configured TmpDir space
24002	Sorter	Error	Sort failed due to insufficient physical TmpDir space
24003	Sorter	Error	Sort failed due to insufficient sort buffer space
24004	Sorter	Error	Sort failed due to too long row (internal failure)
24005	Sorter	Error	Sort failed due to I/O error
30803	Sorter	Error	Illegal value specified for parameter: <parameter>=<value>(legal range is <value>)
30804	Sorter	Error	Sorter temporary directory: <value> does not exist

solidDB RPC errors and messages

Table 60. solidDB RPC errors and messages

Code	Class	Type	Description
21500	RPC	Error	Illegal Ping RPC sequence number. A message was either lost or duplicated.
21501	RPC	Error	Corrupted Ping message.
21502	RPC	Error	Incomplete Ping message. Part of the data was lost.
21503	RPC	Error	Extra bytes in Ping message or header corrupted.
21504	RPC	Error	Requested Ping level is not currently allowed in server. Start listening with -p<*ping level*> option.
21505	RPC	Error	Illegal Ping buffer size or message corrupted.
21506	RPC	Error	Ping session was disconnected abnormally because of a communication error.
21507	RPC	Return Code	Ping test <*ping level*> successful. Results are in file <filename>.
21508	RPC	Error	Ping feature is not supported in the server. Update your server.

Table 60. solidDB RPC errors and messages (continued)

Code	Class	Type	Description
21509	RPC	Error	Failed to write to file <file_name>.
21510	RPC	Error	Failed to read from file <file_name>.
30600	RPC	Message	Received an illegal freearray size <value>
30601	RPC	Message	Received an illegal attribute count <value> routine <value>
30602	RPC	Message	Received an illegal relop <value> routine <value>
30603	RPC	Message	Received an illegal table name <value> routine <value>
30604	RPC	Message	Received an illegal selflags size <value> routine <value>
30605	RPC	Message	Current cursor id <value> found from free array
30606	RPC	Message	Illegal cursor id <value> found from free array
30607	RPC	Message	Received an illegal user id <value>
30608	RPC	Message	Received an illegal connect id <value>
30609	RPC	Message	Received an illegal sequence number <value> expected <value>
30610	RPC	Message	Received an illegal cursor id <value>
30611	RPC	Message	Illegal attribute id <value> in order list
30612	RPC	Message	Illegal attribute id <value> in constraint list
30613	RPC	Message	Illegal attribute id <value> in select list
30614	RPC	Message	Received an illegal length parameter <value> routine <value>
30615	RPC	Message	Received an illegal attribute number parameter routine <value> nattrs <value>
30616	RPC	Message	Cannot send UNICODE string to old client version
30617	RPC	Message	Received an illegal type number routine <value> types <value>
30618	RPC	Message	Received an illegal date attribute from Java client routine <value>
30619	RPC	Message	Received an illegal attribute type parameter routine <value> type <value>
30620	RPC	Message	Received a corrupted data tuple routine <value> row length mismatch
30621	RPC	Message	Received an illegal SQL cursor sync array size <value>
30622	RPC	Message	Received an illegal SQL cursor id <value>in sync array
30623	RPC	Message	Illegal RPC console information
30624	RPC	Message	Illegal RPC session
30625	RPC	Message	Received an illegal done array size <value>
30626	RPC	Message	Received an illegal SQL statement id <value> routine <value>
30627	RPC	Message	Received an illegal SQL statement id <value> pos <value> routine <value>
30628	RPC	Message	Received an illegal read BLOB id <value> routine <value>
30629	RPC	Message	Received an illegal SQL read BLOB buffer size <value> routine <value>
30630	RPC	Message	BLOB data crc failed block count = <value> routine <value>
30631	RPC	Message	Received an illegal BLOB id <value> routine <value>
30632	RPC	Message	Received an illegal BLOB piece length <value> routine <value>
30633	RPC	Message	Received an illegal data length routine <value> length <value>
30634	RPC	Message	Illegal tuple position <value>
30635	RPC	Message	Hot Standby received an illegal counter data size <value> from another server
30636	RPC	Message	Received an illegal replication type parameter <value>
30637	RPC	Message	Ping client from <value> connected
30638	RPC	Message	Ping client from <value> disconnected
30639	RPC	Message	Received an illegal cursor id <value>
30640	RPC	Message	<Server RPC error message>

solidDB synchronization errors

Table 61. solidDB synchronization errors

Code	Class	Type	Description
25001	Synchronization	Error	Master cannot save propagated statements. The master received propagated transaction statements from the replica, but is not able to save the statements. (Note that the master must save the statements before executing them). Possible causes of the error are: • Master database has exceeded the database size limit. You can increase the database size by changing the FileSpec parameter setting in the `solid.ini` file. For details on this parameter, read "FileSpec_[1...n] parameter" on page 50. Be sure to restart the server for the new setting to take effect. • An internal error exists in the database server. If error 25001 occurs even after you have increased the database size, contact IBM Corporation Technical support at http://www.ibm.com/software/data/soliddb/support/.
25002	Synchronization	Error	Cannot save data dictionary statements.

25003	Synchronization	Error	Cannot save SAVE statements.
			
			It is not possible to save a "SAVE" statement for later propagation. For example, the following SQL statement returns an error:
			`SAVE CALL MYPROC(1, 'foo')`
			
			solidDB statements that return this error:
			`SAVE sql_statement`

25004	Synchronization	Error	Dynamic parameters are not supported.
			
			Input parameters of a subscription must be given as literals. They cannot be dynamically bound to the statement.
			
			solidDB statements that return this error:
			`DROP SUBSCRIPTION`
			`MESSAGE message_name APPEND REFRESH publication_name`

25005	Synchronization	Error	Message *message_name* is already active.
			
			A message of the specified name that was created appears to still be active. A message becomes active when the following MESSAGE command is executed:
			`MESSAGE message_name BEGIN`
			
			The message is automatically deleted when the reply of the message has been successfully executed in the replica database.
			
			solidDB statements that return this error:
			`MESSAGE message_name APPEND`
			`MESSAGE message_name BEGIN`
			`MESSAGE message_name DELETE`
			`MESSAGE message_name EXECUTE`
			`MESSAGE message_name FORWARD`
			`MESSAGE GET REPLY`

Table 61. solidDB synchronization errors (continued)

Code	Class	Type	Description
25006	Synchronization	Error	Message *message_name* not active A message has already been committed or ended using the MESSAGE END statement. New tasks cannot be appended to the message using the MESSAGE APPEND command. Probable cause for this error is that the AUTOCOMMIT mode is used in the connection. You must first remove the message with MESSAGE *message_name* DELETE command. Then switch autocommit off and run the script again. solidDB statements that return this error: `MESSAGE message_name APPEND synchronization_task`

25007	Synchronization	Error	Master *master_name* not found
			
			A replica attempts to perform an operation to a master database that cannot be found.
			
			solidDB statements that return this error:
			`SET SYNC CONNECT connect_string TO MASTER master_name`
			`DROP MASTER master_name`
			`IMPORT 'filename'`
			`SAVE sql_statement`

25009	Synchronization	Error	Replica *replica_name* not found
			
			The replica name specified in a command cannot be found.
			
			solidDB statements that return this error:
			`DROP REPLICA replica_name`
			`DROP SUBSCRIPTION publication_name(parameter_list)`
			`[FROM REPLICA replica_name]`
			`GRANT REFRESH ON publication_name`
			`MESSAGE DELETE CURRENT TRANSACTION`
			`MESSAGE message_name [FROM REPLICA replica_name] DELETE`

25010	Synchronization	Error	Publication *publication_name* not found.
			
			The publication name of a subscription is incorrect.
			
			solidDB statements that return this error:
			`MESSAGE APPEND REFRESH publication_name(parameter_list)`
			`DROP PUBLICATION publication_name`
			`EXPORT SUBSCRIPTION publication_name ...`
			`REVOKE REFRESH ON publication_name...`

25011	Synchronization	Error	Wrong number of parameters to publication *publication_name*.
			
			A subscription to a publication contains incorrect number of parameters. The data types of the given subscription parameters must match the input parameter definition of the publication.
			
			solidDB statements that return this error:
			`DROP SUBSCRIPTION publication_name (parameter_list)`
			`[FROM REPLICA replica_name]`
			`MESSAGE message_name APPEND REFRESH`
			`publication_name (parameter_list)`

Table 61. solidDB synchronization errors (continued)

Code	Class	Type	Description

25012	Synchronization	Error	Message reply timed out.
			
			A reply message has not arrived to the replica database within the given timeout period. The reason is that the reply message is not yet ready in the master database. The message needs to be retrieved later using "MESSAGE message_name GET REPLY" command.
			
			solidDB statements that return this error:
			`MESSAGE message_name FORWARD TIMEOUT timeout_in_seconds`
			`MESSAGE message_name GET REPLY TIMEOUT timeout_in_seconds`

25013	Synchronization	Error	Message name message_name not found.
			
			The message with the given name does not exist. The message name is given when the message is created with command MESSAGE message_name BEGIN. The message name is released when the reply message has been successfully executed in the replica database.
			
			Message names must be unique within the replica database.
			
			A message can be deleted from the database with command:
			`MESSAGE message_name [FROM REPLICA replica_name ] DELETE`
			
			solidDB statements that return this error:
			`MESSAGE message_name APPEND`
			`MESSAGE message_name DELETE`
			`MESSAGE message_name END`
			`MESSAGE message_name EXECUTE`
			`MESSAGE message_name FORWARD`
			`MESSAGE message_name FROM REPLICA EXECUTE`
			`MESSAGE message_name FROM REPLICA`
			`replica_name DELETE CURRENT TRANSACTION`
			`MESSAGE message_name GET REPLY`

25014	Synchronization	Error	More than one master name found.

25015	Synchronization	Error	Syntax error: *error_message*, line *line_number*
			
			Syntax is not correct.
			
			solidDB statements that return this error:
			`MESSAGE message_name APPEND`
			`CREATE PUBLICATION publication_name`
			
			Note: See the CREATE PUBLICATION syntax reference for correct syntax.

25016	Synchronization	Error	Message not found, replica id *replica_id*, message id *message_id*
			
			Message not found in master during processing. This can happen if the message is explicitly deleted in master.
			
			solidDB statements that return this error:
			`MESSAGE message_name FORWARD`
			`MESSAGE message_name GET REPLY`
			`MESSAGE message_name RESTART`

Table 61. solidDB synchronization errors (continued)

Code	Class	Type	Description
25017	Synchronization	Error	No unique key found for table *table_name*. The primary key for the table has not been defined. Each table that is part of an incremental publication must have a primary key defined. The synchronization history mechanism cannot function without explicitly defined primary keys. solidDB statements that return this error: `ALTER TABLE table_name SET SYNCHISTORY`

25018	Synchronization	Error	Illegal message state.
			
			An internal error has occurred in the message processing. It is not possible to continue executing the message after this error. Delete the message using the following command:
			`MESSAGE message_name [FROM REPLICA replica_name ] DELETE`
			
			solidDB statements that return this error:
			`MESSAGE message_name ...`

25019	Synchronization	Error	Database is not a replica.
			
			A synchronization message can only be created in a database that has been registered to be a replica database. See the example code in *IBM solidDB Advanced Replication User Guide*, which provides information on registering a replica database.
			
			solidDB statements that return this error:
			`DROP MASTER master_name`
			`DROP PUBLICATION publication_name REGISTRATION`
			`DROP SUBSCRIPTION publication_name ...`
			`IMPORT 'filename'`
			`MESSAGE message_name BEGIN`
			`MESSAGE message_name ENDSET SYNC CONNECT`
			`'connect_string' TO MASTER master_name`

25020	Synchronization	Error	Database is not a master.
			
			A command that can be executed only in a master database has been attempted to execute in a non-master database.
			
			A database can be set to be a master database of a system by entering the following command:
			`SET SYNC MASTER YES`
			
			solidDB statements that return this error:
			`ALTER USER replica_user SET MASTER master_name USER`
			`MESSAGE message_name FROM REPLICA replica_name RESTART`
			`MESSAGE message_name FROM REPLICA replica_name DELETE`
			`DROP REPLICA replica_name DROP SUBSCRIPTION`
			`subscription_name FROM REPLICA replica_name`

Table 61. solidDB synchronization errors (continued)

Code	Class	Type	Description

25021	Synchronization	Error	Database is not master or replica database.
			
			In order to create or drop publication definitions or set the SYNCHISTORY property of a table, the database must be defined to be either master or replica (or both).
			
			solidDB statements that return this error:
			`CREATE PUBLICATION publication_name ...`
			`DROP PUBLICATION publication_name REGISTRATION`
			`SET SYNC MAINTENANCE MODE ...;`
			`ALTER TABLE table_name SET SYNCHISTORY`

25022	Synchronization	Error	User generated error.
			
			The execution of a transaction has been cancelled and rolled back in the master database. Because of the failed transaction, the execution of the message that contained the transaction has been stopped.
			
			User can request solidDB to roll back a transaction by setting the following parameters to the bulletin board of the transaction:
			`PutParam('SYS_ROLLBACK', 'YES')`
			`PutParam('SYS_ERROR_CODE', numeric_value_as_string)`
			`PutParam('SYS_ERROR_TEXT', error_text_as_string)`
			
			If the SYS_ERROR_CODE parameter is not specified or it contains an invalid value, the error number 25022 is returned.
			
			solidDB statements that return this error:
			`MESSAGE message_name FORWARD TIMEOUT timeout_in_seconds`
			`MESSAGE message_name GET REPLY TIMEOUT timeout_in_seconds`

25023	Synchronization	Error	Replica registration failed.
			
			An error has occurred during replica registration.
			
			solidDB statements that return this error:
			`MESSAGE message_name FORWARD TIMEOUT timeout_in_seconds`
			`MESSAGE message_name GET REPLY TIMEOUT timeout_in_seconds`

25024	Synchronization	Error	Master not defined.
			
			No definition for the master exists or the configuration changed during message processing. solidDB was unable to properly initialize the synchronization environment. You can check the master from the replica's system table SYS_SYNC_MASTERS. All successfully registered replicas are found from the master database system table SYS_SYNC_REPLICAS.
			
			Note that this error can be produced if you use double quotes rather than single quotes around the *master_connect_string* in a MESSAGE FORWARD command.
			solidDB statements that return this error:
			`IMPORT 'filename'`
			`MESSAGE message_name FORWARD TO 'master_connect_string'`
			`TIMEOUT timeout_in_seconds`
			
			Note: The use of double quotes rather than single quotes around the *master_connect_string* in can produce this error message.
			`MESSAGE message_name GET REPLY ...`
			`MESSAGE message_name APPEND REFRESH publication_name`
			`MESSAGE message_name EXECUTE ....`

Table 61. solidDB synchronization errors (continued)

Code	Class	Type	Description

25025	Synchronization	Error	Node name not defined.
			
			Before setting up a master database or registering a replica database, the node name of the database must be set. This can be done with the following command:
			`SET SYNC NODE node_name`
			
			solidDB statements that return this error:
			`DROP PUBLICATION publication_name REGISTRATION`
			`MESSAGE message_name APPEND REGISTER REPLICA`
			`MESSAGE message_name BEGIN ...`

25026	Synchronization	Error	A user who has not been defined in the master database, attempts to perform a solidDB SQL command.
			
			solidDB statements that return this error:
			`IMPORT 'filename'`
			`SAVE sql_statement`
			`MESSAGE message_name...`
			
			To resolve this problem, use the correct user ID if there is one. If there is not already a correct user ID, then you have two options:
			
			1) Map a master user to the replica userid you are using. (The master user must already have been downloaded from the master to the replica.) To map a master user to a replica user, execute the command:
			`ALTER USER replica_user SET MASTER master_name`
			`USER user_specification`
			
			2) Add an appropriate user to the master database, and download it with:
			`MESSAGE message_name APPEND SYNC_CONFIG`

25027	Synchronization	Error	Too long column or parameter value; configured maximum is <value>
25028	Synchronization	Error	Message *message_name* can include only one system subscription. System subscriptions (REGISTER REPLICA and SYNC_CONFIG) must be kept in separate messages. These tasks must be the only ones of their messages. solidDB statements that return this error: `MESSAGE message_name APPEND REFRESH publication_name`

25030	Synchronization	Error	Replica *replica_name* is already registered.
			
			A replica attempts to register itself using a name that is already in use. Replica names must be unique. If you know that the chosen replica name is no longer used by any other replicas, drop it from the master database with the command DROP REPLICA replica_name. Then register the replica again. Otherwise, change the newly created replica's name and register it again. Note that replica registration occurs after the registration message is sent to the master.
			
			solidDB statements that return this error:
			`MESSAGE message_name FORWARD ...`
			`MESSAGE message_name GET REPLY ...`

Table 61. solidDB synchronization errors (continued)

Code	Class	Type	Description

25031	Synchronization	Error	Transaction is active, operation failed.
			
			A replica attempts to process a message when having an active transaction.
			
			solidDB statements that return this error:
			`IMPORT 'filename'`
			`MESSAGE message_name FORWARD ...`
			`MESSAGE message_name GET REPLY TIMEOUT ...`
			`MESSAGE message_name EXECUTE`

25032	Synchronization	Error	All publication SQL statements must return rows. The publication definition contains SQL operations that don't return rows. Only SELECT statements are allowed in the publication. solidDB statements that return this error: `CREATE PUBLICATION publication_name`
25033	Synchronization	Error	Publication *publication_name* already exists. A publication has been attempted to create with a name that is already in use. solidDB statements that return this error: `CREATE PUBLICATION publication_name`
25034	Synchronization	Error	Message name *message_name* already exists. Each message must have a name that is unique within the database. solidDB statements that return this error: `MESSAGE message_name BEGIN`

25035	Synchronization	Error	Message *message_name* is in use.
			
			A solidDB message is locked during an attempt to execute it or delete it. A locked message cannot be re-executed or deleted. If you get this error while attempting to create a new solidDB message, it is probably due to an existing message with the same name. You can check existing messages from the system table SYS_SYNC_REPLICA_MSGINFO in the replica or from the system table SYS_SYNC_MASTER_MSGINFO in the master database.
			
			solidDB statements that return this error:
			`MESSAGE message_name BEGIN`
			`MESSAGE message_name END`
			`MESSAGE message_name EXECUTE ...`
			`MESSAGE message_name FROM REPLICA replica_name DELETE`
			`MESSAGE message_name FORWARD TIMEOUT ...`
			`MESSAGE message_name GET REPLY TIMEOUT ...`

Table 61. solidDB synchronization errors (continued)

Code	Class	Type	Description

25036	Synchronization	Error	Publication *publication_name* not found or publication version mismatch.
			
			A publication has been dropped or redefined at master during message processing. Recover by DROP SUBSCRIPTION at replica.
			
			solidDB statements that return this error:
			`IMPORT 'filename'`
			`MESSAGE message_name FORWARD TIMEOUT ...`
			`MESSAGE message_name GET REPLY TIMEOUT ...`
			`MESSAGE message_name EXECUTE ...`

25037	Synchronization	Error	Publication column count mismatch in table *table_name*.
			
			Database definitions at master and replica do not match.
			
			solidDB statements that return this error:
			`MESSAGE message_name FORWARD TIMEOUT timeout_in_seconds`
			`MESSAGE message_name GET REPLY TIMEOUT timeout_in_seconds`
			`MESSAGE message_name EXECUTE`

25038	Synchronization	Error	Table is referenced in publication *publication_name*; drop or alter operations are not allowed.
			
			A table which is referenced in a publication can not be dropped or altered.
			
			solidDB statements that return this error:
			`DROP TABLE table_name`
			`ALTER TABLE table_name`

25039	Synchronization	Error	Table is referenced in subscription to publication *publication_name*; drop or alter operations are not allowed. solidDB statements that return this error: `ALTER TABLE table_name`

25040	Synchronization	Error	User id *user_id* is not found.
			
			User information has been changed at the replica during message execution.
			
			solidDB statements that return this error:
			`IMPORT 'filename'`
			`MESSAGE message_name GET REPLY TIMEOUT timeout_in_seconds`
			`MESSAGE message_name EXECUTE ...`
			`MESSAGE message_name FORWARD ...`

25041	Synchronization	Error	Subscription to publication *publication_name* not found.
			
			The subscription that is expected to be in the replica is not found. This error occurs if the subscription is explicitly dropped at the replica.
			
			solidDB statements that return this error:
			`IMPORT 'filename'`
			`MESSAGE message_name EXECUTE ...`
			`MESSAGE message_name FORWARD ...`
			`MESSAGE message_name GET REPLY ...`
			`DROP SUBSCRIPTION subscription_name`
			`DROP SUBSCRIPTION subscription_name REPLICA replica_name`

Table 61. solidDB synchronization errors (continued)

Code	Class	Type	Description
25042	Synchronization	Error	Message is too long (*number* bytes) to forward. Maximum is set to *number* bytes. The length of a message to be forwarded exceeds the limit for message's length. The limit can be set by variable SYS_R_MAXBYTES_OUT. solidDB statements that return this error: `MESSAGE message_name FORWARD`
25043	Synchronization	Error	Reply message is too long (*number* bytes). Maximum is set to *number* bytes. The length of a message to be received as a reply exceeds the limit for message's length. The limit can be set by variable SYS_R_MAXBYTES_IN. solidDB statements that return this error: `MESSAGE message_name GET REPLY`
25044	Synchronization	Error	SYNC_CONFIG system publication takes only character arguments. In a subscription attempt, publication SYNC_CONFIG was found to have invalid data types for the arguments. solidDB statements that return this error: `MESSAGE message_name APPEND REFRESH SYNC_CONFIG`
25045	Synchronization	Error	Master/replica node support disabled.
25046	Synchronization	Error	Commit and rollback are not supported in propagated transactions. This error is caused when a transaction attempts to execute a COMMIT or ROLLBACK command in the master database. The error is returned to the solidDB server running the procedure. The message containing the procedure will fail.
25047	Synchronization	Error	Parameter info publication not found.

25048	Synchronization	Error	Publication *publication_name* request info not found.
			
			A publication has been dropped while message is being executed.
			
			solidDB statements that return this error:
			`IMPORT 'filename'`
			`MESSAGE message_name EXECUTE ...`
			`MESSAGE message_name FORWARD ...`
			`MESSAGE message_name GET REPLY ...`

25049	Synchronization	Error	Referenced table *table_name* not found in subscription hierarchy. A publication has referenced a table which does not exist. solidDB statements that return this error: `CREATE PUBLICATION publication_name ...`
25050	Synchronization	Error	Table has no history.

Table 61. solidDB synchronization errors (continued)

Code	Class	Type	Description
25051	Synchronization	Error	Unfinished messages found. Replica mode has been attempted to be switched off while there are messages either waiting to be forwarded or being executed at master. solidDB statements that return this error: `SET SYNC REPLICA NO`
25052	Synchronization	Error	Failed to set node name to *node_name*. The *node_name* may be invalid.
25053	Synchronization	Error	Replica not registered in master.

25054	Synchronization	Error	Table *table_name* is not set for synchronization history.
			
			A table in the master database has the SYNCHISTORY property set, but the corresponding table in the replica does not.
			
			solidDB statements that return this error:
			`IMPORT 'filename'`
			`MESSAGE message_name GET REPLY ...`
			`MESSAGE message_name FORWARD ...`

25055	Synchronization	Error	Connect information is allowed only when not registered. The connect info in `MESSAGE message_name FORWARD TO connect_info options` is allowed only if the replica has not yet been registered to the master database. solidDB statements that return this error: `MESSAGE message_name FORWARD TO connect_info options`

25056	Synchronization	Error	Autocommit not allowed.
			
			The solidDB statement must be executed with autocommit mode turned off.
			
			solidDB statements that return this error:
			`All MESSAGE message_name ... statements`
			`DROP SUBSCRIPTION subscription_name`
			`DROP SUBSCRIPTION subscription_name REPLICA replica_name`
			`DROP REPLICA replica_name`
			`DROP MASTER master_name`
			`EXPORT SUBSCRIPTION`
			`IMPORT 'filename'`

25057	Synchronization	Error	Already registered to master *master_name*.
			
			The replica database has already been registered to a master database.
			
			solidDB statements that return this error:
			`MESSAGE message_name GET REPLY ...`
			`(when registering a replica)`
			`MESSAGE message_name FORWARD ...`
			`(when registering a replica)`

25058	Synchronization	Error	Missing connect information.

Table 61. solidDB synchronization errors (continued)

Code	Class	Type	Description
25059	Synchronization	Error	After registration nodename cannot be changed. The SYNC NODE NAME property of a database cannot be changed if the master has any registered replicas or replica has already been registered to a master database. solidDB statements that return this error: `SET SYNC NODE NAME unique_node_name`

25060	Synchronization	Error	Column *column_name* does not exist on publication *publication_name* resultset in table *table_name*.
			
			This error occurs when a replica finds out that the master is transferring data that does not include primary key values that the replica requires.
			
			solidDB statements that return this error:
			`IMPORT 'filename'`
			`MESSAGE message_name GET REPLY ...`
			`MESSAGE message_name FORWARD ...`

25061	Synchronization	Error	Where condition for table *table_name* must refer to an outer table of the publication. If a publication contains nested SELECTs, the WHERE clause of the inner SELECT must refer to the outer table of the outer SELECT. solidDB statements that return this error: `CREATE PUBLICATION publication_name`
25062	Synchronization	Error	User *user_id* is not mapped to master *user_id*. Dropping the user mapping failed because user is not mapped to a given master. solidDB statements that return this error: `ALTER USER replica_user SET MASTER master_name USER`
25063	Synchronization	Error	User *user_id* is already mapped to master *user_id*. User is already mapped to a given master. solidDB statements that return this error: `ALTER USER replica_user SET MASTER master_name USER`
25064	Synchronization	Error	Unfinished message *message_name* found for replica *replica_name*. Dropping the replica failed because there are unfinished messages. solidDB statements that return this error: `DROP REPLICA replica_name`

Table 61. solidDB synchronization errors (continued)

Code	Class	Type	Description
25065	Synchronization	Error	Unfinished message *message_name* found for master *master_name*. Dropping the master failed because there are unfinished messages. solidDB statements that return this error: `DROP MASTER master_name`
25066	Synchronization	Error	Synchronization bookmark *bookmark_name* already exists. Cannot create synchronization bookmark since the name already exists. solidDB statements that return this error: `CREATE SYNC BOOKMARK`
25067	Synchronization	Error	Synchronization bookmark *bookmark_name* not found. Bookmark name is not an existing bookmark. solidDB statements that return this error: `DROP SYNC BOOKMARK`
25068	Synchronization	Error	Export file *file_name* open failure. Failed to open export file for EXPORT SUBSCRIPTION. solidDB statements that return this error: `EXPORT SUBSCRIPTION`
25069	Synchronization	Error	Import file *file_name* open failure. Failed to open import file for IMPORT. solidDB statements that return this error: `IMPORT 'filename'`
25070	Synchronization	Error	Statements can be saved only for one master in transaction. Statements cannot be saved for multiple masters in one transaction. solidDB statements that return this error: `SAVE sql_statement`
25071	Synchronization	Error	Not registered to publication *publication_name*. Replica must be registered to a publication before the publication can be refreshed to the replica. solidDB statements that return this error: `DROP PUBLICATION publication_name REGISTRATION MESSAGE message_name APPEND REFRESH publication_name`

Table 61. solidDB synchronization errors (continued)

Code	Class	Type	Description
25072	Synchronization	Error	Already registered to publication *publication_name*. Replica is already registered to a publication. solidDB statements that return this error: `MESSAGE message_name APPEND REGISTER REPLICA`
25073	Synchronization	Error	Export file can have data only from one master.

25074	Synchronization	Error	User definition not allowed for this operation.
			
			Master user attempts to perform synchronization operation, but is denied access in the replica database because the registration user is still the active user. After the registration process, the command SET SYNC username must be set to NONE.
			
			solidDB statements that return this error:
			`SAVE sql_statement`
			`DROP SUBSCRIPTION publication_name (in replica)`
			`MESSAGE message_name APPEND REFRESH publication_name`
			`MESSAGE message_name APPEND PROPAGATE TRANSACTIONS`
			`MESSAGE message_name APPEND REGISTER PUBLICATION`
			`MESSAGE message_name APPEND UNREGISTER PUBLICATION`
			`MESSAGE message_name EXECUTE (in replica)`

25075	Synchronization	Error	Transaction not found.
25076	Synchronization	Error	Only REGISTER REPLICA is allowed in message.
25077	Synchronization	Error	Node name is not valid.
25078	Synchronization	Error	Node name already exists.
25079	Synchronization	Error	Catalog is master and there are registered replicas. Catalog is not dropped.
25080	Synchronization	Error	Catalog is replica and it is registered to a master. Catalog is not dropped.
25081	Synchronization	Error	Subqueries are not allowed in publication definition.
25082	Synchronization	Error	Node name can not be removed if node is master or replica. Node name cannot be set to NONE on a synchronized master and/or replica catalog. solidDB statements that return this error: `SET SYNC NODE NONE`
25083	Synchronization	Error	Commit block can not be used with HotStandby.
25084	Synchronization	Error	Can not save ADMIN COMMAND.
25085	Synchronization	Error	Failed to store blob from message. During synchronization, reading or storing a BLOB (LONG VARCHAR or LONG VARBINARY data) has failed because of an internal error.

Table 61. solidDB synchronization errors (continued)

Code	Class	Type	Description
25086	Synchronization	Error	Cannot save START statement.

25087	Synchronization	Error	Missing connect information for node *node_name*.
			
			There is no connect string in the table sys_sync_replicas for the specified replica. Registering a replica doesn't doesn't automatically add the connect string into that table if you haven't defined it in the replica's `solid.ini`. You should define it as shown below:
			`[Synchronizer]`
			`ConnectStrForMaster=tcp replicahost 1316`

25088	Synchronization	Error	Catalog already in maintenance mode. You have set the mode on already.
25089	Synchronization	Error	Not allowed to set maintenance mode off. Someone else has set the mode on, so you cannot set it off.
25090	Synchronization	Error	Catalog already in maintenance mode. Someone else has set the mode on, so you cannot set it off.
25091	Synchronization	Error	Catalog is not in maintenance mode. You tried to set the mode off when it was not on.

25092	Synchronization	Error	User version strings are not equal in master and replica, operation failed.
			
			When the replica executes either of the following commands:
			`MESSAGE FORWARD`
			`MESSAGE GET REPLY`
			
			The server checks whether the master and replica sync schema version numbers are equal. If the version numbers are not equal, then the server gives this error. (Note: If neither the master nor the replica has set the version number, then you won't get the error message.)

25093	Synchronization	Error	A master database for this replica exists, operation failed. This message is returned when the user either tries to drop a replica catalog which is registered to a master, or tries to execute 'SET SYNC REPLICA NO' when the replica is registered to a master.
25094	Synchronization	Error	Received illegal message part type.
25095	Synchronization	Error	Message execution aborted.

solidDB HotStandby errors

Table 62. solidDB HotStandby errors

Code	Class	Type	Description
14700	HotStandby	Error	Rejected connection, both servers in PRIMARY role. Meaning: Command 'hsb connect' returns this error if both nodes are in same role.
14701	HotStandby	Error	Rejected connection, both servers in SECONDARY role. Meaning: Command 'hsb connect' returns this error if both nodes are in same role.

Table 62. solidDB HotStandby errors (continued)

Code	Class	Type	Description
14702	HotStandby	Error	Operation failed, catchup is active. Meaning: While the servers are performing catchup, you will get this error if you issue any of the following commands on the Primary: 'hsb switch secondary', 'hsb set secondary alone', 'hsb set standalone', 'hsb connect', 'hsb copy' or 'hsb netcopy'. While the servers are performing catchup, you will get this error if you issue any of the following commands on the Secondary: 'hsb switch primary', 'hsb set secondary alone', 'hsb set primary alone', 'hsb set standalone', or 'hsb connect'.
14703	HotStandby	Error	Operation failed, copy is active. Meaning: While the Primary is doing copy or netcopy, the following commands returns this error: 'hsb switch secondary', 'hsb set secondary alone', 'hsb set standalone', 'hsb connect', 'hsb disconnect', 'hsb copy' or 'hsb netcopy'.
14704	HotStandby	Error	HotStandby copy or netcopy is only allowed when primary is in alone state. Meaning: This error is returned if the server is in PRIMARY ACTIVE state and the command 'hsb copy' or 'hsb netcopy' is issued.
14705	HotStandby	Error	Setting to STANDALONE is not allowed in this state. Meaning: If the server is in PRIMARY ACTIVE state and you issue the command 'hsb set standalone', then you will get this message.
14706	HotStandby	Error	Invalid read thread mode for HotStandby, only mode 2 is supported.
14707	HotStandby	Error	Operation not allowed in the STANDALONE state.
14708	HotStandby	Error	Catchup failed, catchup position was not found from log files.
14709	HotStandby	Error	Hot Standby enabled, but connection string is not defined.
14710	HotStandby	Error	Hot Standby admin command conflict with an incoming admin command.
14711	HotStandby	Error	Failed because server is shutting down.
14712	HotStandby	Error	Server is secondary. Use primary server for this operation.

solidDB SSA (SQL API) errors

Table 63. solidDB SSA (SQL API) errors

Error code			Description
25200	SSA	Error	Invalid application buffer type This error is used for the ODBC driver. It is given if signals attempt to use inappropriate buffer type for reading values (such as reading string to integer value). This error is documented into more detail in the ODBC specifications.
25201	SSA	Error	Invalid use of null pointer This error is given, if an invalid parameter - NULL is passed as a statement handle, connection handle, or application buffer.
25202	SSA	Error	Function sequence error This error is given, if an attempt to violate the ODBC function call sequence is made. This can happen, for example, when trying to execute a statement that has not been prepared.
25203	SSA	Error	Invalid transaction operation code This error is given, if an attempt to use an incorrect transaction completion code with the SQLEndTran function (SQL_COMMIT and SQL_ROLLBACK are allowed) is made.

Table 63. solidDB SSA (SQL API) errors (continued)

Error code			Description
25204	SSA	Error	Invalid string or buffer length This error is given, if 0 or any negative buffer size is passed to an ODBC function that requires an application buffer.
25205	SSA	Error	Invalid attribute/option identifier This error is given, if an invalid operation code is passed to the SQLSetPos, SQLDriverConnect, SQLFreeStmt and so on.
25206	SSA	Error	Connection timeout expired
25207	SSA	Error	Invalid cursor state This error is given, for example, if an attempt is made to fetch with a closed cursor.
25208	SSA	Error	String data, right truncated This error is given if a string buffer was not big enough.
25209	SSA	Error	Datetime field overflow This error is given when updating a date or time column with incorrect data.
25210	SSA	Error	COUNT field incorrect This error is given, for example, when trying to pass an extra parameter to an insert statement.
25211	SSA	Error	Invalid descriptor index This error is given, for example, when using 0 or negative value as SQLBindParameter column index.
25212	SSA	Error	Client unable to establish a connection The ODBC client cannot connect to the server.
25213	SSA	Error	Connection name in use This error is given, for example, when trying to reconnect an already connected connection.
25214	SSA	Error	Connection does not exist This error is given, for example, when trying to use a closed or not connected connection.
25215	SSA	Error	Server rejected the connection Transport layer connection to the server has been established, but the server rejects the connection (for example, because it is shutting down).
25216	SSA	Error	Connection switch, some session context may be lost This is a TF-1 specific error. A TF-1 connection has encountered a connection switch. The application must roll back the transaction to restore the connection.

Table 63. solidDB SSA (SQL API) errors (continued)

Error code			Description
25217	SSA	Error	Client unable to establish a primary connection This is a TF-1 specific error. The ODBC driver has not been able to establish connection to the primary server, for example, after an application rolled back a transaction after a failover, or if there is no primary server address in the TF-1 connection string (all the reachable servers are secondary).
25404	SSA	Error	COUNT field incorrect
25406	SSA	Error	Invalid descriptor index
25411	SSA	Error	String data
25416	SSA	Error	Datetime field overflow
25418	SSA	Error	Invalid cursor state
25424	SSA	Error	Invalid application buffer type
25427	SSA	Error	Invalid use of null pointer
25428	SSA	Error	Function sequence error
25429	SSA	Error	Invalid transaction operation code
25432	SSA	Error	Invalid string or buffer length
25434	SSA	Error	Invalid attribute/option identifier
25448	SSA	Error	Connection timeout expired

solidDB COM (communication) messages

Table 64. solidDB COM (communication) messages

Code	Class	Type	Description
30001	COM	Message	User <username> connected, user id <id>, machine id <id>
30002	COM	Message	User <username> connection timed out, user id <id>, machine id <id>
30003	COM	Message	User <username> was disconnected abnormally, user id <id>, machine id <id>
30004	COM	Message	User <username> disconnected, user id <id>, machine id <id>
30005	COM	Message	Admin user <username> connected, user id <id>, machine id <id>
30006	COM	Message	User <username> connected from a remote control, user id <id>, machine id <id>
30007	COM	Message	User <username> transaction idle timed out, user id , machine id <id>
30008	COM	Message	User <username> transaction timed out, user id %d machine id <id>
30009	COM	Message	User <username> tried to connect from <value> with an illegal username or password.
30010	COM	Message	User <username> failed to connect version mismatch. Client version <version>, server version <version>
30011	COM	Message	User <username> failed to connect, collation version mismatch.
30012	COM	Message	User <username> failed to connect, there are too many connected clients.
30013	COM	Message	New connections allowed.
30014	COM	Message	New connections can not be allowed.
30015	COM	Message	No new connections allowed.
30016	COM	Message	Listening of <connect string> started.
30017	COM	Message	Listening of <connect string> stopped.
30018	COM	Message	No valid listening name specified. Exiting from <server_name>.
30019	COM	Message	Cannot start listening
30020	COM	Message	Server is in fatal state no new connections are allowed
30021	COM	Message	Unknown connection recycling XECB.

solidDB SRV (server) errors

Table 65. solidDB SRV errors

Code	Class	Type	Description
30100	SRV	Message	Server shut down by the application.
30101	SRV	Message	Server shut down by either ALT+F4 or kill command
30102	SRV	Message	User <username> issued shut down server command user id <username>
30103	SRV	Message	Server shut down by unknown user (sc==NULL)
30104	SRV	Message	Shutdown aborted; denied by user callback.
30105	SRV	Message	<server_name> is shut down
30106	SRV	Message	Some thread still active wait extra <value> seconds...
30110	SRV	Message	Service <service_name> installed
30111	SRV	Message	Service <service_name> removed
30112	SRV	Message	Install service <service_name> failed! Error code <error_code>
30113	SRV	Message	Remove service <service_name> failed! Error code <error_code>
30114	SRV	Message	Usage for service option: -s{start | install | remove} name exepath [autostart]
30115	SRV	Message	Failed to change the current working directory to <directory_name>
30116	SRV	Message	Current working directory changed to <directory_name>
30117	SRV	Message	<solidDB_version>
30118	SRV	Message	<copyright>
30119	SRV	Message	<startup_time>
30120	SRV	Message	Failed to start the server. Exiting from <value>
30121	SRV	Message	Causing intentionally an access violation...
30122	SRV	Message	Causing intentionally an internal error...
30123	SRV	Message	Exiting server with ADMIN COMMAND 'errorexit <number>'...
30124	SRV	Message	Exiting server with ADMIN COMMAND 'assertexit'...
30125	SRV	Message	Admin command: <command>
30126	SRV	Message	Admin event: <command>
30127	SRV	Message	Invalid license file <license_file>
30128	SRV	Message	Using license file <license_file>
30129	SRV	Message	Signal <value>
30130	SRV	Message	solidDB process has encountered an internal error and is unable to continue normally.
30131	SRV	Message	Command line: <value>
30132	SRV	Message	SS_DEBUG=<value>
30133	SRV	Message	Asynch pingtest completed successfully to <value>.
30134	SRV	Message	Alternate inifile name is too long (>254); parameter ignored.
30140	SRV	Message	The argument following option -x pagedmem:[client:] must be 16 32 or 64 (default: 16)
30141	SRV	Message	Testing system performance.
30142	SRV	Message	Testing was successful.
30143	SRV	Message	Testing failed.
30144	SRV	Message	Server in backupserver mode. Operation refused
30145	SRV	Message	Connect failed illegal user name or password
30146	SRV	Message	Failed to create thread <value>
30147	SRV	Message	HSB enabled server cannot operate without HotStandby license: set **HotStandby.HSBEnabled** to No.

Table 65. solidDB SRV errors (continued)

Code	Class	Type	Description
30148	SRV	Message	<value> option is activated.
30149	SRV	Message	Server emergency shutdown.
30150	SRV	Fatal Error	Server not started. This error is given if the solidDB server cannot be started.
30151	SRV	Message	Database started.
30152	SRV	Message	Memory allocation size has exceeded <value>MB. Current size: <value> butes. Number of allocations: <value>.
30153	SRV	Message	Memory allocation size has fallen below <value>MB. Current size: <value> bytes. Number of allocations: <value>.
30154	SRV	Message	Statement (id: <userid> userid: <type> type: <value>) has allocated <value> bytes of memory SQL: <value>.
30155	SRV	Message	Process size <virtual_size> is <above|below> the <warning_level|limit|low_level> <value>
30156	SRV	Message	Server health check monitoring started.

solidDB DBE (database engine) errors and messages

Table 66. solidDB DBE errors and messages

Code	Class	Type	Description
30200	DBE	Message	Creating a new database.
30201	DBE	Message	Database converted successfully.
30202	DBE	Message	Database already exists.
30203	DBE	Message	Converting database ...
30204	DBE	Message	This database is from an older Solid version. To convert database for use with this version, start server with option -x convert. Please note that after conversion, the database cannot be used with older versions of server anymore.
30205	DBE	Message	New database was not created.
30206	DBE	Message	Database does not exist. Cannot create a new database because the server is not running as a foreground process. To create a new database, start the server as a foreground process with -f option.
30207	DBE	Message	Failed to open the database. Exiting from <server_name>
30208	DBE	Message	Merge not started; denied by user callback.
30209	DBE	Message	Idle merge started <value> keys to remove
30210	DBE	Message	Merge started, <value> keys to remove
30211	DBE	Message	Idle quick merge started
30212	DBE	Message	Quick merge started
30213	DBE	Message	Merge stopped, all keys merged
30214	DBE	Message	Merge stopped, <value> keys merged
30215	DBE	Message	Merge task started, <value> tasks active
30216	DBE	Message	User merging enabled
30217	DBE	Message	Error when converting procedures procedure <procedure_name>
30218	DBE	Message	Quick merge stopped
30220	DBE	Message	Checking database index
30221	DBE	Message	Database index is ok
30222	DBE	Message	Database is in backup server mode. Cannot check the index.
30223	DBE	Message	Testing the database index.

Table 66. solidDB DBE errors and messages (continued)

Code	Class	Type	Description
30224	DBE	Message	Database index has been tested successfully. Database index is ok.
30225	DBE	Message	ERROR! Database index is NOT ok! Check errors from file `ssdebug.log`.
30226	DBE	Message	SOLID Fatal Error: Failed to open the database for testing.
30227	DBE	Message	SOLID Fatal Error: Failed to connect to the database for testing.
30228	DBE	Message	Database file has been reorganized successfully.
30229	DBE	Message	ERROR! Failed to reorganize the database file! Check errors from file `ssdebug.log`.
30230	DBE	Message	Starting roll-forward recovery, please wait ...
30231	DBE	Message	Recovery of <value> transactions successfully completed
30232	DBE	Message	Recovery successfully completed
30233	DBE	Message	Writing IMDB pages to disk. Pages: <value>
30234	DBE	Message	Finished writing IMDB pages to disk. Pages: <value>
30235	DBE	Message	Loading IMDB. Pages: <value>
30236	DBE	Message	Finished loading IMDB. Pages: <value>
30237	DBE	Message	Starting to reorganize and compact the database file.
30240	DBE	Message	Failed to create a new database
30241	DBE	Message	Failed to log on to the database
30242	DBE	Message	Failed to connect, script not executed.
30243	DBE	Message	Failed to open SQL input file
30244	DBE	Message	Script <script_name> failed
30245	DBE	Message	Table <table_name> not found.
30246	DBE	Message	Converting table <table_name>...
30247	DBE	Message	Table <table_name> converted
30248	DBE	Message	No need to convert table <table_name>
30249	DBE	Message	There is a problem opening the database because not all db files defined is the `solid.ini` were found. Please check the configuration. Note that only the file(s) defined with the largest **FileSpec_n** definition(s) should be missing.
30250	DBE	Message	Using SplitMerge
30251	DBE	Message	Starting to recreate the database (delete old database and create a new one).
30252	DBE	Message	Successfully deleted database and logs
30253	DBE	Message	Failed to delete database and/or logs check file permissions.
30254	DBE	Message	Database is a broken HSB copy or netcopy database.
30255	DBE	Fatal Error	Exiting from server (FAKE_DBE_CRASHAFTERCPMARK).
30256	DBE	Fatal Error	Database must exist!
30257	DBE	Fatal Error	Database creation date is already reset!
30258	DBE	Fatal Error	Database creation time can be reset only once!
30259	DBE	Fatal Error	Error test in file <file_name> line <value>
30260	DBE	Message	Database version does not match with SOLID version.
30261	DBE	Message	Database file format does not match with SOLID version.
30320	DBE	Message	Logreader using default transaction batch size <value>
30321	DBE	Message	Logreader transaction batch size <value>
30322	DBE	Message	Logreader read full statements

Table 66. solidDB DBE errors and messages (continued)

Code	Class	Type	Description
30323	DBE	Message	Logreader catchup init
30324	DBE	Message	Logreader catchup error
30325	DBE	Message	Logreader catchup scan open
30326	DBE	Message	Logreader catchup active
30327	DBE	Message	Logreader catchup completed
30328	DBE	Message	Logreader live data

solidDB CP (checkpoint) messages

Table 67. solidDB CP (checkpoint) messages

Code	Class	Type	Description
30280	CP	Message	Checkpoint creation completed
30281	CP	Message	Checkpoint creation started
30282	CP	Message	Checkpoint creation not started because shutdown is in progress
30283	CP	Message	Checkpoint creation not started because checkpointing is disabled
30284	CP	Message	Checkpoint not started; denied by user callback.
30285	CP	Message	Create <value> start failed.
30286	CP	Message	Checkpoint DBE flushing timed out, <number> of <number> pages left.
30287	CP	Message	Checkpoint MME flushing timed out, <number> of <number> pages left.
30288	CP	Message	MME flush batch completion wait timed out, trying to proceed.
30289	CP	Message	Checkpoint DBE flush, <number> pages left.
30290	CP	Message	Checkpoint MME flush, <number> pages left.

solidDB BCKP (backup) messages

Table 68. solidDB BCKP (backup) messages

Code	Class	Type	Description
30300	BCKP	Message	Backup completed successfully
30301	BCKP	Message	Backup started to <directory path>.
30302	BCKP	Message	Backup start failed. <Shutdown is in progress | Backup is already active>
30303	BCKP	Message	Backup aborted.
30304	BCKP	Message	Backup failed. <error description>
30305	BCKP	Message	Backup not started; denied by user callback.
30306	BCKP	Message	Backup not started; Backup is not supported on diskless server.
30307	BCKP	Message	Backup not started index check failed. Errors written to file `ssdebug.log`.

solidDB AT (timed commands) messages

Table 69. solidDB AT (timed commands) messages

Code	Class	Type	Description
30350	AT	Message	At: backup <backup_directory>
30351	AT	Message	At: makecp

Table 69. solidDB AT (timed commands) messages (continued)

Code	Class	Type	Description
30352	AT	Message	At: throwout <user_name>
30353	AT	Message	At: report <report_file_name>
30354	AT	Message	At: shutdown
30355	AT	Message	At: system <operating_system_command>
30356	AT	Message	At: open
30357	AT	Message	At: close
30358	AT	Message	At: assert
30359	AT	Message	Server noticed time inconsistency during at-command execution. If the system time has been changed, please restart server.
30360	AT	Message	AT command failed. <reason>
30361	AT	Message	Illegal at command <command> ignored.
30362	AT	Message	Illegal immediate at command <command> ignored.
30362	AT	Message	Deleted %d rows from SYS_BACKGROUNDJOB_INFO

solidDB LOG (logging) messages

Table 70. solidDB LOG (logging) messages

Code	Class	Type	Description
30400	LOG	Message	Transaction logging is disabled roll-forward recovery is not possible
30401	LOG	Message	Using log write mode
30402	LOG	Message	Conflicting parameters **General.BackupCopyLog=Yes** and **General.CheckpointDeleteLog=Yes**
30403	LOG	Message	Log file write failure
30404	LOG	Message	Check results from file <file_name>.
30405	LOG	Message	Unable to open message log file <file_name>
30406	LOG	Message	SOLID Fatal Error: Failed to open trace file <file_name>.
30407	LOG	Message	The tail of log was corrupt the corrupt part was ignored.

solidDB INI (configuration file) messages

Table 71. solidDB INI (configuration file) messages

Code	Class	Type	Description
30450	INI	Message	Value <value> for parameter <parameter> is not multiple of 512 using default value <value>
30451	INI	Message	Value for index file specification <specification> is invalid using default file <file_name> and max size <value>
30452	INI	Message	Value for index file specification <specification> is invalid all following file specifications are ignored
30453	INI	Message	Illegal value <value> for parameter <parameter> using default value <value>
30454	INI	Message	Failed to save configuration file <configuration_file>
30455	INI	Message	Failed to set the maximum number of open files to <value> using default <value>
30456	INI	Message	Using configuration file <configuration_file>
30457	INI	Message	Configuration file <configuration_file> not found using defaults
30458	INI	Message	Illegal value <value> for parameter <parameter> using default value <value>
30459	INI	Message	Illegal value <value> for parameter <parameter> using default value <value>

Table 71. solidDB INI (configuration file) messages (continued)

Code	Class	Type	Description
30460	INI	Message	Illegal value <value> for parameter <parameter>using default value <value>
30461	INI	Message	Illegal value <value> for parameter <parameter> using default value <value>
30463	INI	Message	ReadThreadMode forced to (<value>) for parameter <parameter>
30464	INI	Message	Illegal value <value> for parameter <parameter> using default value <value>
30465	INI	Message	Process size <value> exceeds parameter **Srv.ProcessMemoryLimit** value <value> Increase the size of the value of the **Srv.ProcessMemoryLimit** parameter or disable the checking of the process memory size by setting **Srv.ProcessMemoryCheckInterval** parameter value to 0.

solidDB HSB (HotStandby) errors and messages

Table 72. solidDB HSB errors and messages

Code	Class	Type	Description
14007	HSB	Message	CONNECTING
14008	HSB	Message	CATCHUP
14009	HSB	Message	No role switches since the server startup
14010	HSB	Message	DISCONNECTING
14522	HSB	Message	HotStandby copy directory not specified.
14537	HSB	Message	BROKEN
14704	HSB	Error	HotStandby copy or netcopy is only allowed when primary is in alone state
14712	HSB	Error	Server is secondary. Use primary server for this operation
30500	HSB	Message	Started as a HotStandby primary
30501	HSB	Message	Started as a HotStandby secondary
30502	HSB	Message	The database was not shut down properly the last time that it was used starting as a HotStandby secondary
30503	HSB	Message	Forcing HotStandby primary to start as a secondary
30504	HSB	Message	HotStandby role switched to secondary
30505	HSB	Message	HotStandby role switched to primary
30506	HSB	Message	Primary server must be set to PRIMARY ALONE or switched to the secondary role.
30507	HSB	Message	HotStandby server set to PRIMARY ALONE.
30508	HSB	Message	HotStandby server set to SECONDARY ALONE
30509	HSB	Message	HotStandby switch to primary failed, error <error_code>
30510	HSB	Message	HotStandby switch to secondary failed, error <error_code>
30511	HSB	Message	Failed to start HotStandby to <server_name>, error <error_code>
30512	HSB	Message	Failed to switch HotStandby role to primary, error <error_code>
30513	HSB	Message	Failed to switch HotStandby role to secondary, error <error_code>
30514	HSB	Message	Both databases are primary servers starting as a secondary
30515	HSB	Message	Both HotStandby databases are primaries
30516	HSB	Message	Failed to start HotStandby to <server_name>, other server rejected with error <error_code>
30517	HSB	Message	HotStandby role in secondary switched
30518	HSB	Message	HotStandby role switched to standalone
30530	HSB	Message	Starting to send HotStandby catchup data to secondary server
30531	HSB	Message	HotStandby catchup completed successfully
30532	HSB	Message	HotStandby catchup ended abnormally

Table 72. solidDB HSB errors and messages (continued)

Code	Class	Type	Description
30533	HSB	Message	HotStandby catchup can not be started. Secondary is not properly synchronized with primary full synchronization is required
30534	HSB	Message	HotStandby catchup ended abnormally, status <error_code>
30535	HSB	Message	HotStandby catchup ended abnormally, error <error_code>
30536	HSB	Message	HotStandby catchup ended abnormally due to a communication error
30537	HSB	Message	HotStandby catchup ended abnormally, secondary returned error <error_code>
30538	HSB	Message	HotStandby catchup size <value> greater than configured maximum size <value>, stopping HotStandby
30539	HSB	Message	File error in HotStandby catchup, stopping HotStandby
30540	HSB	Message	Starting to receive HotStandby catchup data from primary server
30541	HSB	Message	Secondary is not properly synchronized with primary due to a log file corruption. Please restart secondary and execute a HSB netcopy.
30550	HSB	Message	Connection broken to HotStandby secondary server
30551	HSB	Message	Connected to HotStandby
30552	HSB	Message	HotStandby secondary connected
30553	HSB	Message	HotStandby primary connected
30554	HSB	Message	Hot Standby connection broken to Secondary server with an open transaction waiting for the operator to resolve transaction status. Primary server must be set to alone mode or switched to secondary mode.
30555	HSB	Message	HotStandby ping timeout
30556	HSB	Message	Connection broken to HotStandby secondary
30557	HSB	Message	HotStandby databases are not properly synchronized
30558	HSB	Message	HotStandby connection to secondary timed out
30559	HSB	Message	HotStandby connection broken
30560	HSB	Message	HotStandby: <HotStandby_error_message>
30570	HSB	Message	Network backup completed.
30571	HSB	Message	Started to receive network backup.
30572	HSB	Message	Database started using a HotStandby copy/netcopy.
30573	HSB	Message	Network backup failed.
30574	HSB	Message	Hot Standby forcing threads to 1
30575	HSB	Message	Hot Standby replication configured but no active license found replication not started
30577	HSB	Message	HotStandby connect operation failed
30579	HSB	Message	HotStandby connection is already active.
30581	HSB	Message	Invalid event <event>
30582	HSB	Message	HotStandby cannot set the server to PRIMARY ALONE.
30583	HSB	Message	HotStandby copy failed.
30585	HSB	Message	Database starts to listen for netcopy.
30750	HSB	Message	HotStandby connection is already active.
30752	HSB	Message	Operation failed disconnect is active.
30757	HSB	Message	CONNECTED
30758	HSB	Message	Bad Hot Standby command.
30759	HSB	Message	HotStandby server is set to STANDALONE.
30760	HSB	Message	Started the process of disconnecting the servers.
30761	HSB	Message	Started the process of switching the role to primary.
30762	HSB	Message	Started the process of switching the role to secondary.
30763	HSB	Message	Started the process of connecting the servers.

Table 72. solidDB HSB errors and messages (continued)

Code	Class	Type	Description
30764	HSB	Message	Copy started.
30765	HSB	Message	Parameter AutoPrimaryAlone is set to Yes.
30766	HSB	Message	Parameter AutoPrimaryAlone is set to No.
30767	HSB	Message	Parameter Connect is set to <value>.
30768	HSB	Message	HotStandby connection is already broken.
30769	HSB	Message	Operation failed because connection between the servers is active.
30772	HSB	Message	Hot Standby node identifier must be defined in the ini file.
30774	HSB	Message	Server is already STANDALONE.
30775	HSB	Message	Parameter CopyDirectory is set to <value>.
30776	HSB	Message	Parameter ConnectTimeout is set to <value>.
30777	HSB	Message	Parameter PingTimeout is set to <value> milliseconds.
30779	HSB	Message	Hot Standby migration is active
30782	HSB	Message	Server is already set to primary alone.
30783	HSB	Message	Server is already set to secondary alone.
30784	HSB	Message	Parameter <parameter_name> is set to <value>.
30785	HSB	Message	Parameter <parameter_name> is set to <value>.
30786	HSB	Message	Parameter <parameter_name> is set to <value>.
30787	HSB	Fatal Error	pri_dologskip:bad type, log pos, log size This error refers to a failed operation on the HSB primary server. The error returns the failed operation and its location in the log, and the log size. Operations in the replication log are skipped.
30788	HSB	Fatal Error	pri_hsblogcopy_write:bad type, log pos, log size This error refers to a failed operation on the HSB primary server. The write to the replication log file fails. The error returns the failed operation and its location in the log, and the log size.
30789	HSB	Fatal Error	Failed to open hot standby replication log file.
30790	HSB	Fatal Error	Failed to allocate memory for HotStandby log. Max Log size is *logsize*. This error concerns a diskless database using hotstandby. In these systems, the hotstandby log is written to memory. This error is given if allocating more memory for the log file fails.
30791	HSB	Fatal Error	HotStandby:solhsby:bad type <type>, log pos <log_pos>, log size <log_size>
30792	HSB	Message	Both servers are secondary.

solidDB SNC (synchronization) messages

Table 73. solidDB SNC (synchronization) messages

Code	Class	Type	Description
30700	SNC	Message	Starting parallel sync history key conversion...
30701	SNC	Message	Starting sync history key conversion...
30702	SNC	Message	Sync history key conversion done
30703	SNC	Message	Database is not a master database

solidDB XS (external sorter) errors and messages

Table 74. solidDB XS (external sorter) errors

Code	Class	Type	Description
30800	XS	Message	Unable to reserve requested <number> memory blocks for external sorter. Only <number> memory blocks were available. SQL: <sql statement>
30801	XS	Message	Unable to reserve requested <number> memory blocks for external sorter. Only <number> memory blocks were available.
30802	XS	Fatal Error	Failed to create a temporary file for local sorting (system errno =) The sorter cannot create a temporary file.

solidDB FIL (file system) messages

Table 75. solidDB FIL (file system) messages

Code	Class	Type	Description
30900	FIL	Message	SsBLock failed, file <file_name>, error = <error_code>
30901	FIL	Message	SsBLock failed, file <file_name>, error = <error_code>, fd = <value>
30902	FIL	Message	SsBOpenLocal failed, file <file_name>, error = <error_code>, retries = <value>, open files = <value>
30903	FIL	Message	SsBOpenLocal failed, file <file_name>, error = <error_code>, vaxc$error = <value>, fab stv = <value>, retries = <value>, open files = <value>
30904	FIL	Message	SsBOpenLocal failed, file <file_name>, error = <error_code>, vaxc$error = <value>, retries = <value>
30905	FIL	Message	SsBOpenLocal failed, file <file_name>, error = <error_code>, dos rc = <value>, retries = <value>
30906	FIL	Message	SsBOpenLocal failed, file <file_name>, error = <error_code>, retries = <value>
30907	FIL	Message	SsBOpen failed, file <file_name>, error = <error_code>, retries = <value>
30908	FIL	Message	File flush failed, error <error_code>, file <file_name>
30909	FIL	Message	File flush failed, error <error_code>, vaxc$error = <value>, file <file_name>
30910	FIL	Message	File flush failed, error <error_code>, dos rc <value>, file <file_name>
30911	FIL	Message	File flush close failed, error <error_code>, file <file_name>
30912	FIL	Message	File flush open failed, error <error_code>, file <file_name>
30913	FIL	Message	File size query failed, error<error_code>, file <file_name>, retries <value>
30914	FIL	Message	File size query seek failed, file <file_name>
30915	FIL	Message	File size change failed, error <error_code>, file <file_name>, newsize <value>, retries <value>
30916	FIL	Message	File <file_name>size change failed, not supported by Windows mmio
30917	FIL	Message	File read failed, error <error_code>, file <file_name>, location <directory>, retries <value>
30918	FIL	Message	File read failed, error <error_code>, file <file_name>, location <directory>, retries <value>, vaxc$error = <value>
30919	FIL	Message	File read seek failed, error <error_code>, file <file_name>, location <directory>, retries <value>
30920	FIL	Message	File read seek failed, error <error_code>, file <file_name>, location <directory>, retries <value>, vaxc$error = <value>
30921	FIL	Message	File write failed, error <error_code>, file <file_name>, location <directory>, retries <value>
30922	FIL	Message	File write failed, error <error_code>, file <file_name>, location <directory>, retries <value>, vaxc$error = <value>
30923	FIL	Message	File write seek failed, error <error_code>, file <file_name>, location <directory>, retries <value>
30924	FIL	Message	File write seek failed, error <error_code>, file <file_name>, location <directory> retries <value>, vaxc$error = <value>

Table 75. solidDB FIL (file system) messages (continued)

Code	Class	Type	Description
30925	FIL	Message	File write end failed, error <error_code>, file <file_name>, retries <value>
30926	FIL	Message	File write end failed, error <error_code>, file <file_name>, retries <value>, vaxc$error = <value>
30927	FIL	Message	File append write failed, error <error_code>, file <file_name>, retries <value>
30928	FIL	Message	File append write failed, error <error_code>, file <file_name>, retries <value>, vaxc$error = <value>
30929	FIL	Message	File append seek failed, error <error_code>, file <file_name>, retries <value>
30930	FIL	Message	File append seek failed, error <error_code>, file <file_name>, retries <value>, vaxc$error = <value>
30931	FIL	Message	File seek failed, error <error_code>, file <file_name>, location <directory>, retries <value>
30932	FIL	Message	File seek failed, disk full, error <error_code>, file <file_name>, location <directory>, new location <directory>, retries <value>
30933	FIL	Message	File seek end failed, error <error_code>, file <file_name>, retries <value>
30934	FIL	Message	File seek to new size failed, error <error_code>, file <file_name>, newsize <value>
30935	FIL	Message	File expand write failed, file <file_name>
30936	FIL	Message	File expand seek failed, file <file_name>
30937	FIL	Message	VirtualAlloc failed, error = <error_code>
30938	FIL	Message	File paged read failed, error <error_code>, file <file_name>, npages <value>, pagesize <value>, page address <value>, retries <value>
30939	FIL	Message	File paged write failed, error <error_code>, file <file_name>, npages <value>, pagesize <value>, page address <value>, retries <value>

solidDB TAB (table) messages

Table 76. solidDB TAB (table) messages

Code	Class	Type	Description
31000	TAB	Message	Bad cursor state, function <function> state <state>
31001	TAB	Message	Table <table_name> created as <table_name>

solidDB SQL errors

Table 77. solidDB SQL errors

Error code	Description
SQL Error 1	Parsing error 'syntax error' The SQL parser could not parse the SQL string. Check the syntax of the SQL statement and try again.
SQL Error 2	Table *table* can not be opened You may not have privileges to access the table and its data.
SQL Error 3	Table *table* can not be created Table can not be created. You may not have privileges for this operation.

Table 77. solidDB SQL errors (continued)

Error code	Description
SQL Error 4	Illegal type definition *column* A column type in your CREATE TABLE statement is illegal. Use a legal type for the column.
SQL Error 5	Table *table* can not be dropped Table can not be dropped. Only the owner (that is, the creator) can drop it.
SQL Error 6	Illegal value specified for column *column* The value specified for column is invalid. Check the value for the column.
SQL Error 7	Insert failed The server failed to do the insertion. You may not have INSERT privilege on the table or it may be locked.
SQL Error 8	Delete failed The server failed to do the deletion. You may not have DELETE privilege on the table or the row may be locked.
SQL Error 9	Row fetch failed The server failed to fetch a row. You may not have SELECT privilege on the table or there may be an exclusive lock on the row.
SQL Error 10	View *view* can not be created You cannot create this view. You may not have SELECT privilege on one or more tables in the query-specification of your CREATE VIEW statement.
SQLError 11	View *view* cannot be dropped. You cannot drop this view. Only the owner (i.e. the creator) of the view can drop it.
SQLError 12	Illegal view definition *view* The view definition is illegal. Check the syntax of the definition.
SQLError 13	Illegal column name *column* Column name is illegal. Check that the name is not a reserved name.
SQL Error 14	Call to function *function* failed Function call to function failed. Check the arguments and their types.
SQL Error 15	Arithmetic error An arithmetical error occurred. Check the operators, values and types.
SQL Error 16	Update failed The server failed to update a row. There may a lock on a row.

Table 77. solidDB SQL errors (continued)

Error code	Description
SQL Error 17	View is not updatable This view is not updatable. UPDATE, INSERT and DELETE operations are not allowed.
SQL Error 18	Inserted row does not meet check option condition You tried to insert a row, but one or more of the column values do not meet column constraint definition.
SQL Error 19	Updated row does not meet check option condition You tried to update a row, but one or more of the column values do not meet column constraint definition.
SQL Error 20	Illegal CHECK constraint A check constraint given to the table is illegal. Check the types of the check constraint of this table.
SQL Error 21	Insert failed because of CHECK constraint You tried to insert a row, but the values do not meet the check option conditions.
SQL Error 22	Update failed because of CHECK constraint You tried to update a row, but the values do not meet the check option conditions.
SQL Error 23	Illegal DEFAULT value The DEFAULT value for the column given is illegal.
SQL Error 25	Duplicate columns in INSERT column list You have included a column in column list twice. Remove duplicate columns.
SQL Error 26	At least one column definition required in CREATE TABLE You need to specify at least one column definition in a CREATE TABLE statement.
SQL Error 27	Illegal REFERENCES column list There are wrong number of columns in your REFERENCES list.
SQL Error 28	Only one PRIMARY KEY allowed in CREATE TABLE You can use only one PRIMARY KEY in CREATE TABLE.
SQL Error 29	GRANT failed Granting privileges failed. You may not have privileges for this operation.
SQL Error 30	REVOKE failed Revoking privileges failed. You may not have privileges for this operation.
SQL Error 31	Multiple instances of a privilege type You tried to grant privileges to a role or a user. You have included multiple instances of a privilege type in the list of privileges.

Table 77. solidDB SQL errors (continued)

Error code	Description
SQL Error 32	Illegal constant *constant* Illegal constant was found. Check the syntax of the statement.
SQL Error 33	Column name list of illegal length You have entered different number of columns in CREATE VIEW statement to the view and to the table.
SQL Error 34	Conversion between types failed An expression in UPDATE statement has illegal type for a column.
SQL Error 35	Column names not allowed in ORDER BY for UNION You can not use column name in an ORDER BY for UNION statement.
SQL Error 36	Nested aggregate functions Nested aggregate functions can not be used. For example: SUM(AVG(column)).
SQL Error 37	Aggregate function with no arguments An aggregate function was entered with no arguments. For example: SUM().
SQL Error 38	Set operation between different row types You have tried to execute a set operation of tables with incompatible row types. The row types in a set operation must be compatible.
SQL Error 39	COMMIT WORK failed Committing a transaction failed.
SQL Error 40	ROLLBACK WORK failed Rolling back a transaction failed.
SQL Error 41	Savepoint could not be created A savepoint could not be created.
SQL Error 42	Could not create index *index* An index could not be created. You may not have privileges for this operation. You need to be an owner of the table or have SYS_ADMIN_ROLE to have privileges to create index for the table.
SQL Error 43	Could not drop index *index* An index could not be dropped. You may not have privileges for this operation. You need to be an owner of the table or have SYS_ADMIN_ROLE to have privileges to drop index from the table.
SQL Error 44	Could not create schema *schema* A schema could not be created.

Table 77. solidDB SQL errors (continued)

Error code	Description
SQL Error 45	Could not drop schema *schema* A schema could not be dropped.
SQL Error 46	Illegal ORDER BY specification You tried to use an ORDER BY column that does not exist. Refer to an existing column in the ORDER BY specification.
SQL Error 47	Maximum length of identifier is 31 You have exceeded the maximum length for the identifier.
SQL Error 48	Subquery returns more than one row You have used a subquery that returns more than one row. Only subqueries returning one row may be used in this situation.
SQL Error 49	Illegal expression *expression* You tried to insert or update a table using an aggregate function (SUM, MAX, MIN or AVG) as a value. This is not allowed.
SQL Error 50	Ambiguous column name *column* You have referenced a column which exists in more than one table. Use syntax *table.column* to indicate which table you want to use.
SQL Error 51	Non-existent function *function* You tried to use a function which does not exist.
SQL Error 52	Non-existent cursor *cursor* You tried to use a cursor which is not created.
SQL Error 53	Function call sequence error A function was called in wrong order. Check the sequence and success of the function calls.
SQL Error 54	Illegal use of a parameter A parameter was used illegally. For example: SELECT * FROM TEST WHERE ? < ?;
SQL Error 55	Illegal parameter value A parameter has an illegal value. Check the type and value of the parameter.
SQL Error 56	Only ANDs and simple condition predicates allowed in UPDATE CHECK All search condition predicates are not supported.
SQL Error 57	Opening the cursor did not succeed Server failed to open a cursor. You may not have cursor open at this moment.

Table 77. solidDB SQL errors (continued)

Error code	Description
SQL Error 58	Column *column* is not referenced in group-by-clause You tried to group rows using column. All columns in group_by_clause must be listed in your select_list. A star ('*') notation is not allowed with GROUP BY.
SQL Error 59	Comparison between incompatible types You tried to compare values which have incompatible types. Incompatible types are for example an integer and a date value.
SQL Error 60	Reference to the insert table not allowed in the source query You have referenced in subquery a table where you are inserting values. This is not allowed.
SQL Error 61	Reference to the update table not allowed in subquery You have referenced in subquery a table where you are updating values. This is not allowed.
SQL Error 62	Reference to the delete table not allowed in subquery You have referenced in subquery a table where you are deleting values. This is not allowed.
SQL Error 63	Subquery returns more than one column You have used a subquery that returns more than one column. Only subqueries returning one column may be used.
SQL Error 64	Cursor *cursor* not updatable The cursor opened is not updatable.
SQL Error 65	Insert or update tried on pseudo column You tried to update a pseudo column (ROWID, ROWVER). Pseudo columns are not updatable.
SQL Error 66	Could not create user *user* A user could not be created. You may not have privileges for this operation.
SQL Error 67	Could not alter user *user* A user could not be altered. You may not have privileges for this operation.
SQL Error 68	Could not drop user *user* A user could not be dropped. You may not have privileges for this operation.
SQL Error 69	Could not create role *role* A role could not be created. You may not have privileges for this operation.
SQL Error 70	Could not drop role *role* A role could not be dropped. You may not have privileges for this operation.

Table 77. solidDB SQL errors (continued)

Error code	Description
SQL Error 71	Grant role failed Granting role failed. You may not have privileges for this operation.
SQL Error 72 Revoke role failed	Revoking role failed. You may not have privileges for this operation.
SQL Error 73	Comparison of vectors of different length You have tried to compare row value constructors that have different number of dimensions. For example you have compared (a,b,c) to (1,1).
SQL Error 74	Expression * not compatible with aggregate expression The aggregate expression can not be used with * columns. Specify columns using their names when used with this aggregate expression. This usually happens when GROUP BY expression is used with the * columns.
SQL Error 75	Illegal reference to table *table* You have tried to reference a table which is not in the FROM list. For example: SELECT T1.* FROM T2.
SQL Error 76	Ambiguous table name *table* You have used the syntax *table.column_name* ambiguously. For example: SELECT T1.* FROM T1 A,T1 B WHERE A.F1=0;
SQL Error 77	Illegal use of aggregate expression You tried to use aggregate expression illegally. For example: SELECT ID FROM TEST WHERE SUM(ID) = 3;
SQL Error 78	Row fetch failed The server failed to fetch a row. You may not have SELECT privilege on the table or there may be an exclusive lock on the row.
SQL Error 79	Subqueries not allowed in CHECK constraint You tried to use subquery in a check constraint.
SQL Error 80	Sorting failed External sorter is out of disk space or cache memory. Modify parameters in configuration file `solid.ini`.
SQL Error 81	SET syntax results in error
SQL Error 82	Improper type used with LIKE
SQL Error 83	Syntax error
SQL Error 84	Parser error *statement*
SQL Error 85	Incorrect number of values for INSERT

Table 77. solidDB SQL errors (continued)

Error code	Description
SQL Error 86	Illegal ROWNUM constraint
SQL Error 88	Subquery not allowed in UPDATE expression Subqueries cannot be used with UPDATE statements.
SQL Error 90	Incorrect ALTER table
SQL Error 93	Illegal GROUP BY expression GROUP BY expression is illegal.
SQL Error 102	Unused optimizer hint A table name alias was used in the query, but this alias was not specified as the table name in the optimizer hint. The alias name must be specified, not the table name.

solidDB executable errors

Table 78. solidDB executable errors

Error code	Description
Executable Error 10	Failed to open database
Executable Error 11	Failed to connect to database
Executable Error 12	Database test failed
Executable Error 13	Database fix failed
Executable Error 14	License error
Executable Error 15	Database must be converted
Executable Error 16	Database does not exist
Executable Error 17	Database exists
Executable Error 18	Database not created
Executable Error 19	Database create failed
Executable Error 20	Communication init failed
Executable Error 21	Communication listen failed
Executable Error 22	Service operation failed
Executable Error 23	Failed to open all the defined database files.
Executable Error 24	Database is a broken netcopy database
Executable Error 50	Illegal command line argument
Executable Error 51	Failed to change directory

Table 78. solidDB executable errors (continued)

Error code	Description
Executable Error 52	Input file open failed
Executable Error 53	Output file open failed
Executable Error 54	Server connect failed
Executable Error 55	Operation init failed
Executable Error 100	Assert or other fatal error.

solidDB Speed Loader (solload) errors

Table 79. solidDB Speed Loader (solload) errors

Error Code	Meaning
No error code	Operation was successful
No error code	Operation has completed
100	Operation failed. For example, this error code is procedured when performing an operation, such as flushing arrays and inserting records.
106	Illegal column name This error applies to the column name used in the control file.
107	Illegal constraint
108	Invalid column data The data type in the data file conflicts with the table definition.
109	Unique constraint violation
110	Concurrency conflict, two transactions updated or deleted the same row
112	Unsupported character set
114	Null data in NOT NULL column NULL data value given in a NOT NULL column
116	Communication error, connection is lost
121	RPC parameter error
122	Table not found
124	Wrong number of parameters

Appendix E. solidDB ADMIN COMMAND syntax

This appendix describes the solidDB ADMIN COMMAND syntax. This command set is not part of ANSI SQL; it is a solidDB-specific extension.

ADMIN COMMAND

```
ADMIN COMMAND 'command_name'

command_name ::= ABORT | ASSERTEXIT | BACKUP |
BACKGROUNDJOB | BACKUPLIST | CHECKPOINTING | CLEANBGJOBINFO |
CLOSE | DESCRIBE | ERRORCODE | ERROREXIT | ERRORMESSAGE | FILESPEC |
HELP | HOTSTANDBY | INDEXUSAGE | INFO | LOGMESSAGE | MAKECP | MEMORY |
MESSAGES | MONITOR | NETBACKUP | NETBACKUPLIST | NETSTAT | NOTIFY |
OPEN | PARAMETER | PERFMON | PERFMON DIFF | PID | PROCTRACE |
PROTOCOLS | REPORT | RUNMERGE | SAVE | SHUTDOWN | SQLLIST | STARTMERGE |
STATUS | THROWOUT | TID | TRACE | TRACEMESSAGE | USERID | USERLIST |
USERTRACE | VERSION
```

Usage

The ADMIN COMMAND is a solidDB-specific SQL extension that executes administrative commands.

Using ADMIN COMMAND with solidDB SQL Editor (solsql)

When used with the solidDB SQL Editor (solsql), the *command_name* must be given with quotes. For example:

```
ADMIN COMMAND 'backup'
```

Using ADMIN COMMAND with solidDB Remote Control (solcon)

When used with the solidDB Remote Control (solcon), the ADMIN COMMAND syntax includes the *command_name* only, without the quotes. For example:

```
backup
```

Abbreviations

Abbreviations for ADMIN COMMANDs are also available. For example:

```
ADMIN COMMAND 'bak'
```

To access a list of abbreviated commands, execute

```
ADMIN COMMAND 'help'
```

The result set contains two columns: RC and TEXT:

- The RC (return code) column is a command return code. If the execution of the command was successful, value 0 is returned.
- The TEXT column is the command reply.

Important usage notes

- **All options of the ADMIN COMMAND are not transactional and cannot be rolled back.**
- **ADMIN COMMANDs and starting transactions**

Although ADMIN COMMANDs are not transactional, they will start a new transaction if one is not already open. (They do not commit or roll back any open transaction.) This effect is usually insignificant. However, it may affect the 'start time" of a transaction, and that may occasionally have unexpected effects. solidDB's concurrency control is based on a versioning system; you see a database as it was at the time that your transaction started.

For example, if you issue an ADMIN COMMAND without another commit and then leave for an hour; when you return, your next SQL command may see the database as it was an hour ago, that is, when you first started the transaction with the ADMIN COMMAND.

- **Error codes**

  Error codes in ADMIN COMMANDS return an error only if the command syntax or parameter values are incorrect. If only the requested operation may be started, the command returns SQLSUCCESS (0). The outcome of the operation itself is written into a result set. The result set has two columns: RC and TEXT. The RC (return code) column contains the return code of the operation: it is "0" for success, and different numeric values for errors. It is thus necessary to check both the codes of the ADMIN COMMAND statement and of the operation.

Following is a description of the syntax for each ADMIN COMMAND command option:

Table 80. ADMIN COMMAND syntax and options

Option syntax	Description
`ADMIN COMMAND 'abort [backup | netbackup]'`	Aborts the active local or network backup process. The backup operation is not guaranteed to be atomic, therefore the cancelled operation may produce an incomplete backup file to the backup directory until the next backup takes place. If the option is not entered, the command defaults to `ADMIN COMMAND 'abort backup'`.
`ADMIN COMMAND 'assertexit'` `Abbreviation: asex`	Terminates the server immediately without a proper shut down.

`ADMIN COMMAND 'backgroundjob' [LIST [-l] [user]] | [ABORT {jobid | user | ALL }] | [DELETE ERRORINFO {jobid | user | ALL }]'`
 	
`user ::= USER {username|userid}`
 	
`Abbreviation: bgjob`
	Lists and possibly aborts running background jobs, that is, SQL statements that have been started by using the `START AFTER COMMIT` statement.
	• LIST option lists running jobs for all users or a specified user.
	• -l option refers to a long list (similar to `ADMIN COMMAND 'userlist -l'`).
	• ABORT option aborts either jobs by job identification number or all jobs by user identification number. If you give the ABORT without arguments, it aborts all jobs from all users.
	• DELETE ERRORINFO option deletes error information from the SYS_BACKGROUNDJOB_INFO system table, where the errors encountered by background jobs are stored. This option performs the same operation as the deprecated `ADMIN COMMAND 'CLEANBGJOBINFO'` command.

`ADMIN COMMAND 'backup [-s] [backup_directory]'` `Abbreviation: bak`	Makes a backup of the database. The operation can be performed as a synchronized or an asynchronic (default) manner. The synchronized operation is specified by using the optional -s option. The default backup directory is defined with the **General.BackupDirectory**. The backup directory may also be given as an argument. For example, `backup abc` creates a backup in directory `abc`. All directory definitions are relative to the solidDB working directory.
`ADMIN COMMAND 'backuplist'` `Abbreviation: bls`	Displays a status list of last local backups.
`ADMIN COMMAND 'checkpointing {ON|OFF}'` `Abbreviation: cp`	Turns checkpointing on or off.

Table 80. ADMIN COMMAND syntax and options (continued)

Option syntax	Description
`ADMIN COMMAND 'cleanbgjobinfo'` `Abbreviation: cleanbgi`	**Note:** This command has been deprecated. Use `ADMIN COMMAND 'backgroundjob'` instead. Cleans the table SYS_BACKGROUNDJOB_INFO containing status data of background procedures.
`ADMIN COMMAND 'close'` `Abbreviation: clo`	Closes the server to new connections; no new connections are allowed.

`ADMIN COMMAND 'describe parameter param'`
`Abbreviation: des`
	Returns a description of all parameters or a parameter specified with *param*.
	
	param must be given in the format **section_name.param_name**. The section and parameter names are case-insensitive.
	
	The following example describes parameter **Com.Trace = y/n**:
	`ADMIN COMMAND 'des parameter com.trace'  RC TEXT`
	`-- ----`
	`0 Trace`
	`0 If set to 'yes', trace information of the network messages is written to a file`
	`0 BOOL`
	`0 RW/STARTUP`
	`0`
	`0`
	`0 No`
	`7 rows fetched.`

`ADMIN COMMAND 'errorcode {all | SOLID_error_code}'`
`Abbreviation: ec`
	Returns a description of all error codes or a specific error code.
	
	SOLID_error_code is the code number, for example `10034`.
	`ADMIN COMMAND 'errorcode 10034';`
	`RC TEXT`
	`-- ----`
	`0 Code: DBE_ERR_SEQEXIST (10034)`
	`0 Class: Database`
	`0 Type: Error`
	`0 Text: Sequence already exists`
	`4 rows fetched.`

`ADMIN COMMAND 'errorexit <number>'` `Abbreviation: erex`	Forces the server into an immediate process exit with the given process exit code.
`ADMIN COMMAND 'errormessage <string>'` `Abbreviation: errmsg`	Outputs the user-defined `<string>` to the error message log (`solerror.out`).
`ADMIN COMMAND 'filespec'` `Abbreviation: fs`	Displays database file specifications defined with the **IndexFile.FileSpec** parameter as well file sizes and current fill ratios (percentage).
`ADMIN COMMAND 'help'` `Abbreviation: ?`	Displays available commands.
`ADMIN COMMAND 'hotstandby [option]'` `Abbreviation: hsb`	A HotStandby command. For a list of options, see the *IBM solidDB High Availability User Guide*. For a list of options, see HotStandby ADMIN COMMANDs in the *IBM solidDB High Availability User Guide*.
`ADMIN COMMAND 'indexusage'` `Abbreviation: idxu`	Displays the indexes, showing the number of times each index has been used.

Table 80. ADMIN COMMAND syntax and options (continued)

Option syntax	Description

`ADMIN COMMAND 'info [options]'`
`Abbreviation: info`
	Returns server information.
	
	The output consists of 25 rows of data.
	
	options are as follows:
	• numusers - Number of current users.
	• maxusers - Maximum number of users.
	• sernum - Server serial number.
	• dbsize - Database size.
	• logsize - Size of log files.
	• uptime - Server up since.
	• bcktime - Timestamp of last successfully completed local backup.
	• cptime - Timestamp of last successfully completed checkpoint.
	• tracestate - Current trace state.
	• monitorstate - Current monitor state, shown as the number of users who have SQL monitoring currently enabled (see `ADMIN COMMAND 'monitor'` for information on SQL monitoring).
	If all users have SQL monitoring enabled, the value is `-1`.
	• openstate - Current open or close state — that is, whether the database server accepts new connections or not. `Open` means that the database server accepts new connections.
	• nummerges - Number of merges.
	• numlocks - Number of locks.
	• numcursors - Number of open cursors.
	• numtransactions - Number of open transactions.
	• memtotal - Total amount of memory allocated bytes.
	• dbfreesize - Amount of free space remaining in database.
	• dbpagesize - Database page size.
	• imdbsize - Amount of space used by in-memory tables (including temporary tables and transient tables) and the indexes on those tables. The return value is in kilobytes (KB) and is in the form of a VARCHAR.
	• name - Server name.
	• primarystarttime - The time the Primary role has started.
	• secondarystarttime- The time the Secondary role has started.
	• dbconfigsize - The configured database size.
	• dbcreatetime - This option prints out the database creation timestamp. The abbreviation dbcreationtime can also be used.
	• processsize - This option prints out the system-level virtual process size in kilobytes. The abbreviation psize can also be used.
	
	More than one option can be used per command. Values are returned in the same order as requested, one row for each value.
	
	Example:
	`ADMIN COMMAND 'info dbsize logsize';`
	`RC TEXT`
	`-- ----`
	`0 851968`
	`0 573440`
	`2 rows fetched.`

`ADMIN COMMAND 'logmessage <string>'` `Abbreviation: logmsg`	Outputs the user-defined `<string>` to the message log (`solmsg.out`).

Table 80. ADMIN COMMAND syntax and options (continued)

Option syntax	Description
`ADMIN COMMAND 'makecp [-s]'` `Abbreviation: mcp`	Makes a checkpoint. Only users with SYS_ADMIN_ROLE privilege can execute this command. By default, the checkpoint is asynchronous. With the option -s, the command returns only after the checkpoint has completed.
`ADMIN COMMAND 'memory'` `Abbreviation: mem`	Returns the server process memory size. The reported process memory size can differ from the process size reported by your operating system.

`ADMIN COMMAND 'messages`
`[{ warnings | errors}] [`*`count`*`]'`
`Abbreviation: mes`
	Displays server messages. Optional severity and message numbers can also be defined. For example:
	
	ADMIN COMMAND 'messages warnings 100' displays last 100 warnings.

`ADMIN COMMAND 'monitor`
`{on | off} [ user`
`{`*`username`* `|` *`userid`*`}]'`
`Abbreviation: mon`
	Sets server monitoring on and off.
	
	When set to `on`, user activity and SQL calls are logged into the `soltrace.out` file.

`ADMIN COMMAND 'netbackup`
`[`*`options`*`] [DELETE_LOGS |`
`KEEP_LOGS] [connect`
`connect str``] [dir`
`backup dir``]'`
`Abbreviation: nbak`
	Makes a network backup of the database. The operation can be performed as a synchronized or an asynchronic (default) manner. The synchronized operation is specified by using the -s option.
	
	DELETE_LOGS means that backed-up log files in the source server are deleted. This is sometimes referred to as *full backup*. This is the default value.
	
	KEEP_LOGS means that backed-up log files are kept in the source server. This is sometimes referred to as *copy backup*. Using KEEP_LOGS corresponds to setting the **General.NetbackupDeleteLog** parameter to no.
	
	The default connect string and the default netbackup directory are defined with the **General.NetBackupConnect** and the **General.NetBackupDirectory** parameters.
	
	The options that are entered with this command override the values specified in the configuration file.
	
	Directory definitions are relative to the solidDB working directory.

`ADMIN COMMAND 'netbackuplist'` `Abbreviation: nbls`	Displays a status list of the most recently made network backups of the database server.
`ADMIN COMMAND 'netstat'` `Abbreviation: net`	Displays server settings and the network status.

`ADMIN COMMAND 'notify`
`user {`*`username`* `|` *`user id`* `|` `ALL }`
`message``'`
`Abbreviation: not`
	This command sends an event to a given user with event identifier NOTIFY. This identifier is used to cancel an event-waiting thread when the statement timeout is not long enough for a disconnect or to change the event registration.
	
	The following example sends a notify message to a user with user id 5 ; the event then gets the value of the message parameter.
	`ADMIN COMMAND 'notify user 5 Canceled by admin'`

`ADMIN COMMAND 'open'` `Abbreviation: ope`	Opens server for new connections; new connections are allowed.

Table 80. ADMIN COMMAND syntax and options (continued)

Option syntax	Description
`ADMIN COMMAND 'parameter [-r][name[= [*|value][temporary]]' Abbreviation: par`	Displays and sets server parameter values. If you run the command without any options, all parameters are displayed. The output can contain three columns. For example: `0 PassThrough SqlPassthroughRead Force Conditional None` • First column shows the current value (`Force`) that might have been changed dynamically. • Second column shows the value set in the `.ini` file at startup. (`Conditional`) • Third column shows the factory value. (`None`) • -r means that only the current parameter values are returned. • name may be a section name or a parameter name prefaced by a section name (**section_name.parameter_name**). There must be a period between the section name and the parameter name. • = [*|value][temporary] – If you assign a parameter value with an asterisk (*), the parameter will be set to its factory value. – If value is not specified, the parameter will be set to its startup value. – temporary means that the changed value is not stored in the `solid.ini` file. For example: • `'parameter general'` displays all parameters from section `[General]`. • `'parameter general.readonly'` displays the parameter **Readonly** in the `[General]` section. • `'parameter com.trace=yes'` sets communication trace on. • `'parameter com.trace='` sets communication trace to its startup value. • `'parameter com.trace=*'` sets communication trace to its factory value.
`ADMIN COMMAND 'perfmon [- c | - r] [print_options] [name_prefix_list]' Abbreviation: pmon`	Returns server performance counters for the past few minutes at approximately one minute intervals. Most values are shown as the average number of events per second. Counters that cannot be expressed as events per second (for example, database size) are expressed in absolute values. • -c - prints actual counter values for each snapshot. • -r - prints counter values in raw mode, which includes only the latest counter values without any formatting. The counter names are not printed. This option is useful if actual monitoring is performed using some other external program that retrieves the counter values from the server. You can retrieve the counter names with the --xnames option. • print_options – -xtime - prints the time in seconds – -xtimediff - prints the difference to the last pmon call in milliseconds – -xnames - prints out the column names for the output – -xdiff - indicates the difference to the last ADMIN COMMAND 'perfmon' execution instead of the absolute value • *name_prefix_list* - limits the output to specific counter types, as indicated by the first word in the counter name. For example, to print all File related counters, the *name_prefix_list* should be `file`. You can also specify multiple prefixes. The following example returns all information: ADMIN COMMAND 'perfmon' The following example returns all values for counters whose name starts with prefix `File` and `Cache`. ADMIN COMMAND 'perfmon -c file cache'

Table 80. ADMIN COMMAND syntax and options (continued)

Option syntax	Description
`ADMIN COMMAND 'perfmon diff [ start | stop ] [filename][interval]' Abbreviation: pmon diff`	Starts a server task that prints out all perfmon counters with specified intervals to a file. • *filename* is the name of the output file. The performance data is output in comma-separated value format; the first row contains the counter names, and each subsequent row contains the performance data per each sampling time. The default file name is `pmondiff.out`. • *interval* is the interval in milliseconds at which performance data is collected. The default interval is 1000 milliseconds. The following command starts a task that outputs performance data to `myd.csv` file on 500 milliseconds interval: ADMIN COMMAND 'pmon diff start myd.csv 500'
`ADMIN COMMAND 'pid' Abbreviation: pid`	Returns server process id.
`ADMIN COMMAND 'proctrace { on | off } user username { procedure | trigger | table } entity_name' Abbreviation: ptrc`	This turns on tracing in stored procedures and triggers. *username* is the name of the user whose procedure calls (or triggers) you want to trace. If multiple connections are using the same username, calls from all of those connections will be traced. Furthermore, if you are using advanced replication, the tracing will be done not only for calls on the replica, but also calls that are propagated to the master and then executed on the master. *entity_name*is the name of the procedure, trigger, or table for which you want to turn tracing on or off. If you specify a procedure or trigger name, then it will generate output for every statement in the specified procedure or trigger. If you specify a table name, then it will generate output for all triggers on that table. Trace is activated only when the specified username calls the procedure / trigger. For more details about proctrace, see section Tracing facilities for stored procedures and triggers in *IBM solidDB SQL Guide*. See also `ADMIN COMMAND 'usertrace'`.

`ADMIN COMMAND 'protocols' Abbreviation: prot`	Returns a list of available communication protocols, one row for each protocol.
	Example (Windows environments):
	`ADMIN COMMAND 'protocols';`
	`RC TEXT`
	`-- ----`
	`0 NmPipe np`
	`0 TCP/IP tc`
	`2 rows fetched.`

`ADMIN COMMAND 'report filename' Abbreviation: rep`	Generates a report of server information to a file defined with *filename*.
`ADMIN COMMAND 'runmerge' Abbreviation: rm`	Runs an index merge.
`ADMIN COMMAND 'save parameters [filename]' Abbreviation: save`	Saves the set of current configuration parameter values to a file. If no file name is given, the default `solid.ini` file is rewritten. This operation is performed implicitly at each checkpoint.
`ADMIN COMMAND 'shutdown [force]' Abbreviation: sd`	Stops solidDB. If the force option is used, the active transactions are aborted and the users are disconnected forcefully.
`ADMIN COMMAND 'sqllist top number_of_statements'`	This command prints out a list of the longest running SQL statements among the currently running statements. The list contains the selected number of statements.
`ADMIN COMMAND 'startmerge' Abbrevation: sm`	Starts and waits for completion of merge.

Table 80. ADMIN COMMAND syntax and options (continued)

Option syntax	Description
`ADMIN COMMAND 'status'` `Abbreviation: sta`	Displays server statistics.

`ADMIN COMMAND 'status`
`backup | netbackup'`
`Abbreviation: sta backup | netbackup`
	Displays status of the last started local or network backup. The status can be one of the following:
	• If the last backup was successful or no backups have been requested, the output is 0 SUCCESS.
	• If the backup is in process (for example, started but not ready yet), then the output is 14003 ACTIVE.
	• If the backup is being finalized, the output is 14003 STOPPING.
	• If the last backup failed, the output is: *errorcode* ERROR where the *errorcode* shows the reason for the failure.

`ADMIN COMMAND 'throwout`
`{`*`username`* `|`
` `*`userid`* `| all}'`
`Abbreviation: to`
	Exits all or specific users from solidDB. To exit a specified user, give the username or user id as an argument. To throw out all users, use the keyword ALL as an argument.

`ADMIN COMMAND 'tid'` `Abbreviation: tid`	This command returns the ID (4-digit code) of the current user thread (in the server).

`ADMIN COMMAND 'trace`
`{ on | off } sql | est |`
`estplans | rpc |`
`sync | flowplans |`
`rexec | batch | logreader |`
`info <level> |`
`all | active'`
`Abbreviation: tra`
	Sets server trace on or off.
	The name of the default trace file is `soltrace.out`.
	The tracing options are:
	• sql - SQL messages
	• est - SQL estimator information
	• estplans - SQL execution plan
	• rpc - Network communications
	• sync - synchronization messages
	• flowplans - plans of SQL statements related to advanced replication
	• rexec - remote procedure call information
	• batch - background job and deferred procedure call information
	• logreader - logs the following information into the trace file `soltrace.out`.
	– Logreader read started.
	– Errors in logreader cursor start. Total of 14 different error conditions are printed.
	– Logreader read stopped.
	– Abnormal read stop after certain system changes.
	– High level information of number of returned log records and read progress.
	Each information is tagged with user id so operations from different users can be separated.
	• info <level> - SQL execution trace (level can be 0...8)
	• all - both SQL messages and network communications messages are written to the trace file.
	• active - lists all active traces

`ADMIN COMMAND 'tracemessage <string>'` `Abbreviation: trcmsg`	Outputs the user-defined `<string>` to the trace message log (`soltrace.out`).

Table 80. ADMIN COMMAND syntax and options (continued)

Option syntax	Description

`ADMIN COMMAND 'userid'`
`Abbreviation: uid`
	Returns the user identification number of the current connection.
	
	The lifetime of an Id is that of the user session. After a user logs out, the number may be reused.
	`ADMIN COMMAND 'userid'`
	`RC TEXT`
	`-- ----`
	`0 8`
	`1 rows fetched.`
	
	For example, the userid can be used in the ADMIN COMMAND "throwout' command to disconnect a specific user.

Table 80. ADMIN COMMAND syntax and options (continued)

Option syntax	Description
`ADMIN COMMAND 'userlist [-l] [name | id]'` `Abbreviation: ul`	This command displays a list of users that are currently logged into the database, as well as information about various database operations and settings for each user. The option `-l` (long) displays a more detailed output. Without the `-l` option, the following information is displayed: *User name*, *User Id*, *Type*, *Machine Id*, *Login time*, and *Appinfo* (if available). With the `-l` option, the following information is displayed: • *Id* - The user session identification number (userid) within the database. The lifetime of the userid is that of the user session. After a user logs out, the number may be reused. • *Type* - Client type. Possible values are: – *Java*, which refers to a client using JDBC – *ODBC*, which refers to a client using ODBC – *SQL*, which refers to solidDB SQL Editor (solsql) • *Machine* - The client computer name (host name) and its IP address, if available • *Login tile* - The client computer login timestamp • *Appinfo* - The value of the client computer's environmental variable SOLAPPINFO (ODBC), or the value of JDBC connection property solid_appinfo. • *Last activity* - The time when the client last time sent a request to the server. • *Autocommit* - Value 0 means that the autocommit mode is switched off; the current transaction is open until a COMMIT or ROLLBACK statement is issued. Value 1 means that the autocommit mode is switched on; each statement is automatically committed. • *RPC compression* - Indicates whether the data transmission compression is on or off. • *Transparent failover* - This field indicates if Transparent Failover (TF) is in use (HotStandby configurations). Because solidDB tools do not support TF, you will only see a "no" value in this field when using solsql or solcon. • *Transparent cluster* - Transparent cluster indicates whether the load balancing feature (in HSB) is enabled for this connection or not. • *Transaction active* - This field indicates whether there is an open, uncommitted transaction on the connections (value 1) or not (value 0). When the connection is set for Autocommit, the value is, most of the time, 0. • *Transaction duration* - This field indicates the duration of the currently open transaction. After COMMIT or ROLLBACK, the value becomes 0. • *Transaction isolation* - This field indicates the transaction isolation level for the transactions. The isolation level decides how data which is a part of an ongoing transaction is made visible to other transactions. • *Transaction durability* - This field indicates the durability of the currently open transaction. • *Transaction safeness* - This field indicates the safeness of the currently open transaction (set with **HotStandby.SafenessLevel**). • *Transaction autocommit* - This field indicates whether the currently open transaction is automatically committed. If the transaction autocommit for the current transaction is switched off (value 0), the current transaction is open until a COMMIT or ROLLBACK statement is issued. After that, a new statement starts a new transaction. If the autocommit mode is switched on for the current transaction (value 1), each statement is automatically committed.

Table 80. ADMIN COMMAND syntax and options (continued)

Option syntax	Description

..continued..
`ADMIN COMMAND 'userlist [-l] [name | id]'`
`Abbreviation: ul`
	• *Current catalog* - Indicates the current catalog name.
	• *Current schema* - Indicates the current schema name.
	• *Sortgroubby* - Indicates how the `GROUP BY` statement is performed if explicit information about the number of result groups is not available. There are two possible values:
	– `ADAPTIVE` - `GROUP BY` input is pre-sorted if the real number of result groups exceeds the number of rows that fit into the central memory array for `GROUP BY`.
	– `STATIC` - `GROUP BY` input is pre-sorted whenever there are at least two items in the `GROUP BY` list. Otherwise, the `GROUP BY` input is not pre-sorted.
	• *Simple optimizer rules* - Indicates whether simple optimizer rules are in use (**SQL.SimpleOptimizerRules**) Possible values are Yes/No/Default.
	• *Statement max time* - Indicates the connection-specific statement maximum execution time in seconds. This setting is effective until a new maximum time is given. Zero time indicates that there is no maximum time. This is the default value.
	• *Lock timeout* - Indicates the timeout set by using the `SET LOCK TIMEOUT` statement.
	• *Optimistic lock timeout* - Indicates the timeout set by using the `SET OPTIMISTIC LOCK TIMEOUT` statement.
	• *Idle timeout* - Indicates the timeout set by using the `SET IDLE TIMEOUT` statement.
	• *Join Path Span* - Indicates the join path span value set by using the `SET SQL JOINPATHSPAN` statement.
	• *RPC seqno* - Internal protocol message sequence number.
	• *SQL sortarray* - The size of user-specific internal sort array.
	• *SQL unionsfromors* - The value tells how many (at most) OR operators may be converted to UNIONs. Unions are faster but require more memory to execute.
	• *EVENT QUEUE LENGTH* - Indicates the number of posted events in the event queue.
	• *Connection idle timeout* - Indicates the connection idle timeout setting
	• *Stmt id* - The current statement identification number. The numbers are session specific and they are assigned for each different statement.
	• *Stmt state* - An internal statement execution state.
	• *Stmt rowcount* - The number of rows retrieved or inserted in the current statement.
	• *Stmt start time* - The current statement start date and time.
	• *Stmt last activity time* -
	• *Stmt duration* - Internal statement duration in seconds. Note: this value has no relevance to the externally visible statement latency. Typically, the statement duration is much longer than latency.
	• *Stmt SQL str* - The current SQL statement string.

Table 80. ADMIN COMMAND syntax and options (continued)

Option syntax	Description
`ADMIN COMMAND 'usertrace { on | off } user username { procedure | trigger | table } entity_name'` `Abbreviation: utrc`	This turns on user tracing in stored procedures and triggers. This command will generate output for every WRITETRACE statement in the specified procedure or trigger. • *username* is the name of the user whose procedure calls (or triggers) you want to trace. If multiple connections are using the same username, then calls from all of those connections will be traced. Furthermore, if you are using advanced replication, the tracing will be done not only for calls on the replica, but also calls that are propagated to the master and then executed on the master. • *entity_name* is the name of the procedure, trigger, or table for which you want to turn tracing on or off. If you specify a table name, it will generate output for all triggers on that table. Trace is activated only when the specified user calls the procedure / trigger. For more details about usertrace, see section Tracing facilities for stored procedures and triggers in *IBM solidDB SQL Guide*. See also ADMIN COMMAND 'proctrace'.
`ADMIN COMMAND 'version'` `Abbreviation: ver`	Displays server version information and information related to the solidDB software licence in use.

Index

Special characters

A

B

C

D

E

F

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Notices

This product is protected by U.S. patents 6144941, 7136912, 6970876, 7139775, 6978396, 7266702, 7406489, 7502796, and 7587429.

This product is assigned the U.S. Export Control Classification Number ECCN=5D992b.

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the

names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com®, Solid, solidDB, InfoSphere, DB2®, Informix®, and WebSphere® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other product and service names might be trademarks of IBM or other companies.

Printed in USA

SC23-9824-03