IBM solidDB
Version 7.0

Administrator Guide

SC27-3839-05

Note
Before using this information and the product it supports, read the information in "Notices" on page 383.

First edition, fifth revision

This edition applies to V7.0 Fix Pack 8 of IBM solidDB (product number 5724-V17) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Oy IBM Finland Ab 1993, 2013

Contents

Figures vii

Tables ix

Summary of changes xi

About this manual xv
Typographic conventions xv
Syntax notation conventions xvi

1 Overview of solidDB administration . . 1

2 Administering solidDB 5
2.1 Automated and manual administration 5
2.2 Starting and stopping server 6
2.2.1 Starting solidDB 6
2.2.2 Closing a database (preventing new connections) 11
2.2.3 Shutting down solidDB. 12
2.3 Creating a new database 13
2.3.1 Usernames, passwords, and system catalog names 14
2.3.2 Unicode and partial Unicode database modes 15
2.3.3 Setting up database environment 16
2.3.4 solidDB configuration file (`solid.ini`) . . . 17
2.3.5 Setting database block size (BlockSize) and location (FileSpec) 18
2.3.6 Defining database objects 19
2.4 Connecting to solidDB with solidDB tools (**`solsql`** and **`solcon`**) 20
2.5 Running solidDB server as a Windows service 21
2.5.1 Starting solidDB server as a service for the first time 21
2.5.2 Starting and stopping solidDB services . . 22
2.5.3 Removing solidDB services 23
2.6 Running several servers on one computer . . . 23
2.7 Performing backup and recovery 24
2.7.1 Making local backups 24
2.7.2 Making backups over network 25
2.7.3 Configuring and automating backups . . . 27
2.7.4 What happens during backup 28
2.7.5 Administering network backup server. . . 30
2.7.6 Monitoring and controlling backups . . . 30
2.7.7 Correcting a failed backup 31
2.7.8 Troubleshooting backups 31
2.7.9 Restoring backups 31
2.7.10 Transaction logging 32
2.8 Creating checkpoints 33
2.9 Entering timed commands 34
2.10 Compacting database files (database reorganization) 35

3 Configuring solidDB 37
3.1 Managing parameters 38
3.1.1 Configuration files and parameter settings 38
3.1.2 Viewing and setting parameters with ADMIN COMMAND 39
3.1.3 Setting parameters through the `solid.ini` configuration file 42
3.1.4 Access mode and persistence of parameter modifications 45
3.1.5 Format of configuration parameter names and values. 46
3.1.6 Most important server-side parameters . . 47
3.1.7 Most important client-side parameters. . . 52
3.2 Using solidDB command-line options 54
3.3 Setting environment variables specific to solidDB 55

4 Security 57
4.1 Authentication 57
4.1.1 Default solidDB authentication 57
4.1.2 Operating-system-based external authentication 58
4.2 Encryption 76
4.2.1 Enabling encryption with IBM Global Security Kit (GSKit) 76
4.2.2 Encrypting database and log files 78
4.2.3 Starting an encrypted database 80
4.2.4 Changing the encryption password. . . . 80
4.2.5 Decrypting a database 80
4.2.6 Disabling encryption of passwords 81
4.2.7 Setting RSA key length for GSKit encryption 82
4.2.8 Querying database encryption status . . . 82
4.2.9 Making backups of encrypted databases . . 83
4.2.10 Encrypting HotStandby servers 83
4.2.11 Encryption and performance 83
4.3 Authorization, privileges, and roles 84
4.4 Using solidDB with SELinux 85
4.5 Using solidDB audit trail (**`AuditTrailEnabled`**) 86
4.5.1 Enabling and disabling audit trail 86
4.5.2 Querying audit trail data in the SYS_AUDIT_TRAIL system table 87
4.6 Troubleshooting encryption and authentication 87
4.6.1 solidDB server startup fails with error `External authentication requires GSKit to be enabled or GSKit enabled, but failed to load the GSKit library` 87
4.6.2 Connection for an externally authenticated user fails at SQLAllocEnv 88
4.6.3 Connection for an externally authenticated user fails with `Error 08004: Server rejected the connection` 89
4.6.4 External authentication with Java fails with `java.lang.UnsatisfiedLinkError: ssolidac70` . 89

5 Monitoring solidDB 91
5.1 Viewing error messages and log files. 92

5.1.1 Controlling message log output 92
5.1.2 Viewing error message descriptions with ADMIN COMMAND 'errorcode' 92
5.1.3 Using trace files 93
5.1.4 Tracing failed login attempts 93
5.2 Checking solidDB version 94
5.3 Checking solidDB ODBC and JDBC client version 94
5.4 Checking database status 95
5.5 Obtaining list of connected users 96
5.6 Disconnecting (throwout) connected users . . . 97
5.7 Querying the status of the most recent backup 97
5.8 Producing reports 97
5.9 Performance counters (perfmon) 98
5.9.1 ADMIN COMMAND 'perfmon' 98
5.9.2 ADMIN COMMAND 'perfmon diff' - producing a continuous performance monitoring report . . 99
5.9.3 ADMIN COMMAND 'perfmon timers' 100
5.9.4 List of perfmon counters 102

6 Managing network connections . . . 117
6.1 Communication between client and server . . 117
6.2 Network listening names (Com.Listen) 118
6.2.1 Viewing supported protocols for the server 120
6.2.2 Viewing network names for the server . . 120
6.2.3 Adding and modifying a network name for the server 120
6.2.4 Removing network name from the server 121
6.3 Connect strings for ODBC clients 121
6.3.1 Default connect string (Com.Connect) . . 123
6.3.2 Logical data source names 123
6.4 Connect strings for JDBC clients 125
6.5 Direct linking with shared memory access (SMA) and linked library access (LLA) 125
6.6 Communication protocols 125
6.6.1 TCP/IP protocol 125
6.6.2 UNIX Pipes 127
6.6.3 Named Pipes. 127
6.6.4 Shared Memory 128
6.6.5 Summary of protocols. 128

7 Using solidDB data management tools 131
7.1 solidDB Remote Control (**solcon**) 131
7.1.1 Starting solidDB Remote Control (**solcon**) 132
7.1.2 Entering commands in solidDB Remote Control (**solcon**) 133
7.2 solidDB SQL Editor (**solsql**) 134
7.2.1 Starting solidDB SQL Editor (**solsql**). . . 134
7.2.2 Executing SQL statements with solidDB SQL Editor (**solsql**) 136
7.2.3 Executing an SQL script from a file . . . 137
7.2.4 solidDB SQL Editor (**solsql**) commands 137
7.3 solidDB Speed Loader (**solloado** and **solload**) 138
7.3.1 File types 138
7.3.2 Starting solidDB Speed Loader (**solloado** and **solload**) 140
7.3.3 Tips for speeding up loading 142
7.3.4 Examples of solidDB Speed Loader usage 142
7.3.5 Control file syntax 144
7.4 solidDB Export (solexp). 152
7.4.1 Starting solidDB Export (solexp) 152
7.5 solidDB Data Dictionary (soldd) 155
7.5.1 Starting solidDB Data Dictionary (soldd) 155
7.6 Entering password from a file. 158
7.7 Using solidDB tools with Unicode 158
7.8 Example: Reloading a database using solidDB tools 159

8 Performance tuning 163
8.1 Logging and transaction durability 163
8.1.1 Background 163
8.1.2 Balancing performance and safety. . . . 164
8.1.3 How relaxed transaction durability can improve performance. 165
8.2 Choosing transaction isolation levels 165
8.2.1 Setting the isolation level. 166
8.3 Controlling memory consumption 166
8.3.1 Controlling process size 167
8.3.2 Tuning your operating system 169
8.3.3 Database cache 169
8.3.4 Sorting 171
8.4 Cache segment partitioning 172
8.5 Tuning network messages 175
8.6 Tuning I/O 175
8.6.1 Distributing I/O 175
8.6.2 Setting the MergeInterval parameter . . . 175
8.7 Tuning checkpoints 176
8.8 Reducing Bonsai Tree size by committing transactions 177
8.8.1 Preventing excessive Bonsai Tree growth 177
8.9 Diagnosing poor performance. 179

9 Troubleshooting and support. . . . 181
9.1 Troubleshooting a problem. 181
9.1.1 Tools for troubleshooting. 183
9.1.2 Troubleshooting licensing issues 193
9.1.3 Troubleshooting Universal Cache 194
9.1.4 Troubleshooting SMA 197
9.1.5 Troubleshooting database file size (file write fails) 199
9.1.6 Troubleshooting MME.ImdbMemoryLimit 200
9.1.7 Troubleshooting solidDB Data Dictionary (soldd) 202
9.1.8 Troubleshooting encryption and authentication 202
9.2 Searching knowledge bases 202
9.3 Getting fixes 203
9.4 IBM Software Support for solidDB 204
9.4.1 Contacting IBM Support 204
9.4.2 Collecting diagnostics data 205
9.4.3 Subscribing to Support and other updates 210

Appendix A. Server-side configuration parameters 213
A.1 Accelerator section 213
A.2 Cluster section 214
A.3 Communication section 214
A.4 General section 217
A.5 HotStandby section 228

A.6 IndexFile section 231
A.7 Logging section 234
A.8 LogReader section 238
A.9 MME section 240
A.10 Passthrough section 243
A.11 SharedMemoryAccess section 245
A.12 Sorter section 246
A.13 SQL section 247
A.14 Srv section 251
A.15 Synchronizer section 264

Appendix B. Client-side configuration parameters 267
B.1 Client section 267
B.2 Communication section 268
B.3 Data sources section 269
B.4 SharedMemoryAccess section 270
B.5 TransparentFailover section 270

Appendix C. solidDB command-line options 271

Appendix D. Environment variables 275

Appendix E. Error codes 277
E.1 solidDB system errors 279
E.2 solidDB database errors 282
E.3 solidDB table errors 291
E.4 solidDB session errors 306
E.5 solidDB communication errors 307
E.6 solidDB server errors 310
E.7 solidDB procedure errors 316
E.8 solidDB API errors 319
E.9 solidDB sorter errors 319
E.10 solidDB RPC errors and messages 320
E.11 solidDB synchronization errors 321
E.12 solidDB HotStandby errors 335
E.13 solidDB SSA (SQL API) errors 336
E.14 solidDB COM (communication) messages . . 338
E.15 solidDB SRV (server) errors 339
E.16 solidDB DBE (database engine) errors and messages 341
E.17 solidDB CP (checkpoint) messages 343
E.18 solidDB BCKP (backup) messages 343
E.19 solidDB AT (timed commands) messages . . 343
E.20 solidDB LOG (logging) messages 344
E.21 solidDB INI (configuration file) messages . . 344
E.22 solidDB HSB (HotStandby) errors and messages 345
E.23 solidDB SNC (synchronization) messages . . 347
E.24 solidDB XS (external sorter) errors and messages 348
E.25 solidDB FIL (file system) messages 348
E.26 solidDB TAB (table) messages 349
E.27 solidDB SMA (shared memory access) errors 349
E.28 solidDB PT (passthrough) errors 349
E.29 solidDB SQL errors 350
E.30 solidDB executable errors 356
E.31 solidDB Speed Loader (solloado and solload) errors 357

Appendix F. solidDB ADMIN COMMAND syntax 359
F.1 ADMIN COMMAND 359

Index 375

Notices 383

Figures

Tables

1. Typographic conventions xv
2. Syntax notation conventions. xvi
3. solidDB default files 16
4. Connecting to solidDB 20
5. Options for the **backup** command 25
6. Options for the **netbackup** command 25
7. **ADMIN COMMAND 'backup'** and `solid.ini` parameter correspondence for local backups . 27
8. **ADMIN COMMAND 'netbackup'** and `solid.ini` parameter correspondence for network backups. 27
9. Backup and netbackup commands 30
10. Arguments and defaults for different timed commands 34
11. Connect string options 53
12. solidDB environment variables 55
13. GSKit libraries and default installation locations 59
14. GSKit libraries and default installation locations 61
15. GSKit libraries and default installation locations 63
16. Linked library access (LLA) system libraries 64
17. GSKit libraries and default installation locations 77
18. System roles 84
19. Perfmon counters 103
20. Network listening name options 118
21. **Com.Listen** factory values 120
22. Connect string options 121
23. TCP/IP protocol in the network listening name (**Com.Listen**) 126
24. TCP/IP protocol in the client connect string (**Com.Connect**) 126
25. UNIX Pipes protocol in the network name 127
26. Named Pipes protocol in the network name 127
27. Shared Memory protocol in the network name 128
28. solidDB protocols and network names 128
29. Application protocols and network names 129
30. **solcon** command options 132
31. **solcon** specific commands 133
32. **solsql** command options 134
33. solidDB SQL Editor (**solsql**) commands 137
34. **solloado** and **solload** command options 140
35. Full syntax of the control file 145
36. solidDB Speed Loader reserved words 146
37. Data masks 147
38. solexp command options. 152
39. soldd command line options 155
40. Command line options for solidDB tools for partial Unicode and Unicode databases . . . 159
41. Determining command status 178
42. Determining which connections have committed transactions 178
43. Diagnosing poor performance 179
44. SQL Info levels 184
45. Comparison of the Monitor facility and the SQL Trace facility 189
46. Ping facility levels 192
47. SMA default address spaces. 199
48. **solidsupport** 205
49. solidDB Support Assistant (**solidsupport**) options 207
50. Accelerator parameters 213
51. Cluster parameters. 214
52. Communication parameters. 214
53. General parameters 217
54. HotStandby parameters 228
55. IndexFile parameters 231
56. Logging parameters 234
57. Log Reader parameters 238
58. MME parameters 240
59. SQL passthrough parameters 243
60. Shared memory access parameters 245
61. Sorter parameters 246
62. SQL parameters. 247
63. Srv parameters 251
64. Synchronizer parameters. 264
65. Client parameters 267
66. Client-side communication parameters 268
67. Data Sources parameters. 269
68. Shared memory access parameters (client-side) 270
69. TransparentFailover parameters 270
70. solidDB environment variables. 275
71. solidDB error categories 277
72. solidDB system errors. 279
73. solidDB database errors 282
74. solidDB session errors 306
75. solidDB communication errors 307
76. solidDB server errors 310
77. solidDB SA API errors 319
78. solidDB sorter errors 319
79. solidDB RPC errors and messages 320
80. solidDB synchronization errors. 321
81. solidDB HotStandby errors 335
82. solidDB SSA (SQL API) errors 336
83. solidDB COM (communication) messages 338
84. solidDB SRV errors 339
85. solidDB DBE errors and messages 341
86. solidDB CP (checkpoint) messages 343
87. solidDB BCKP (backup) messages. 343
88. solidDB AT (timed commands) messages 343
89. solidDB LOG (logging) messages 344
90. solidDB INI (configuration file) messages 344
91. solidDB HSB errors and messages 345
92. solidDB SNC (synchronization) messages 347
93. solidDB XS (external sorter) errors 348
94. solidDB FIL (file system) messages 348
95. solidDB TAB (table) messages 349
96. solidDB SMA (shared memory access) errors 349

97. solidDB passthrough errors 349
98. solidDB SQL errors 350
99. solidDB executable errors 356
100. solidDB Speed Loader (solloado and solload) errors 357
101. ADMIN COMMAND syntax and options 360

Summary of changes

Changes for revision 05

- Added new section Cache segment partitioning.
- New parameters introduced in Fix Pack 7 added in section Server-side configuration parameters:
  - **General.RSAKeySize**
  - **HotStandby.NetcopyRpcCompress**
  - **IndexFile.BonsaitreeJoinLimit**
  - **IndexFile.BtreeJoinLimit**
- New error messages for Fix Pack 7 added in section Error codes; see *Changes in solidDB® server between versions 7.0 and 6.5* for details.
- New performance counters introduced in Fix Pack 7 added in section List of perfmon counters:

```
B-tree node join
B-tree node storage fill factor
B-tree node Bonsai fill factor
Storage tree height
```

- Detailed information about **ADMIN COMMAND 'status'** output added in section ADMIN COMMAND.
- A previously undocumented command **ADMIN COMMAND 'getreadonlyflag'** added in section ADMIN COMMAND.
- New parameter **Logging.LogSoftMemoryLimit** introduced in Fix Pack 6 added in section Logging section.
- New performance counters introduced in Fix Pack 6 added in section List of perfmon counters:

```
Log writeq bytes allocated
Log writeq items freed
Log writeq items blocking waits
Log writeq items only distribute

MME ffmem purge
MME ffmem purge step
```

- Error messages 30286 and 30287 updated in section CP (checkpoint) messages.

Changes for revision 04

- New section added:
  - Troubleshooting licensing issues
- New performance counters introduced in Fix Pack 5 added in section List of perfmon counters:

```
Latency below 1 ms
Latency below 2 ms
Latency below 4 ms
Latency below 8 ms
Latency below 16 ms
Latency below 32 ms
Latency below 64 ms
Latency below 128 ms
Latency below 256 ms
Latency below 512 ms
Latency over 512 ms
```

- Updated Synchronizer section with information about whether the advanced replication configuration parameters are used on master, replica or both databases.

Changes for revision 03

- New performance counters added in section List of perfmon counters:

```
Mem page alloc
Mem page free

HSB last catchup recs
```

- New error messages added in section Error codes; see *Changes in solidDB server between versions 7.0 and 6.5* for details.
- New parameter **SQL.PreferExactNumericFunctions** added in section SQL section.

Changes for revision 02

- New parameters added in section Server-side configuration parameters:
  - **LogReader.UseThrottling**
  - **General.NetBackupReceiveBufferSize**
  - **HotStandby.NetcopyReceiveBufferSize**
- New performance counters added in section List of perfmon counters:

```
Parallel fetch
Parallel fetch wait
Parallel threads active
```

- New error messages added in section Error codes; see *Changes in solidDB server between versions 7.0 and 6.5* for details.
- New sections added:
  - Example: Configuring external authentication for JDBC connections - Windows
  - ADMIN COMMAND 'perfmon timers'
- Information about the **Srv.ConnectionCheckInterval** parameter updated in section Srv section: to enable the use of **Srv.ConnectionCheckInterval**, the **Srv.ReadThreadMode** parameter must be set to 0.
- Information about the ADMIN COMMAND 'userlist' command updated in section ADMIN COMMAND. As of 7.0.0.2 Interim 2, the ODBC and JDBC client version is included in the ADMIN COMMAND 'userlist' printouts.
- Parameter changes in section Server-side configuration parameters:
  - New parameters added:
    - **Srv.InifileLineSplitting**
    - **Srv.MaxUsers**
    - **Srv.ReportInterval**
- Factory value for the **Com.SocketLinger** client-side parameter changed from `yes` to `no` in section Communication section.
- New sections with information about how to configure and use external authentication added:
  - Installing and configuring IBM® Global Security Kit (GSKit) for external authentication
  - Installing and configuring IBM Global Security Kit (GSKit) for external authentication – JDBC clients
  - Checking authentication type of users

- Section Querying database encryption status updated: the function DATABASE_ENCRYPTION_LEVEL() returns the database encryption level:
  - 0 – no encryption
  - 1 – encrypted
- New performance counters added in section List of perfmon counters:

```
RPC connected
RPC disconnected
SQL execute simple
SQL execute complex
DBE fetch M-table
DBE fetch D-table
Search replan
Tabcur table scan
Tabcur index access
```

- Previously undocumented client-side parameters **Com.SocketLinger** and **Com.SocketLingerTime** added in section Client-side configuration parameters.
- Information about error message `Server Error 14534: Only administrative statements are allowed.` updated in section solidDB server errors. The error is returned when the solidDB process size has exceeded the limit set with parameter **Srv.ProcessMemoryLimit**.
- The use of quotation marks in ADMIN COMMAND clarified in section ADMIN COMMAND: the *command_name* in ADMIN COMMAND '*command_name*' must be given with single quotation marks.
- Section for client-side parameter **ODBCHandleValidation** corrected; the **ODBCHandleValidation** is in the Client section.

Changes for revision 01

- Information about external authentication and encryption updated in section Security.
- New parameters added in section General section
  - **General.DefaultDomainName**
  - **General.GSKitLoginRequired**
  - **General.PAMServiceName**
- New parameter added in section Client-side configuration parameters
  - **Client.GSKitPath**
- Factory values for the following parameters updated in section Server-side configuration parameters:
  - **SQL.InfoFileSize=100 M** (old value 1 M)
  - **Srv.TraceLogSize=100 M** (old value 1 M)
  - **Srv.TraceSecDecimals=3** (old value 0)
- New error messages added in section Error codes; see *Changes in solidDB server between versions 7.0 and 6.5* for details.

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 solidDB components can be enacted to serve special needs. Such components are:

- In-memory database
- Highly available HotStandby configuration
- Advanced asynchronous replication
- Linked access model where the application is linked directly 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 that the reader has general database management system (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.

Table 1. Typographic conventions (continued)

Format	Used for
argument	Words emphasized like this indicate information that the user or the application must provide.
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 Overview of solidDB administration

The solidDB server is designed for ease of deployment and continuous, unattended operation.

Installation

The solidDB server is delivered as a single installation program file. All the drivers and utilities are included in the installation package.

For details on the installation alternatives and the package contents, see the IBM solidDB Getting Started Guide.

Creating a database

When solidDB is started, it checks if a database exists. If a database does not exist, solidDB prompts you to create a database.

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

Administration interface

The main administration interface for the solidDB server is the **ADMIN COMMAND** SQL extension that is used to executed administrative commands. The ADMIN COMMANDs are used for operations such as creating backups of the database, invoking performance monitoring, or displaying information about users connected to the database. The ADMIN COMMANDs can also be used for changing certain configuration settings dynamically.

The solidDB package includes two command-line utilities that you can use to execute the administrative commands.

solidDB SQL Editor (solsql)
: solidDB SQL Editor (**solsql**) is a console tool that you can use to issue SQL statements and solidDB ADMIN COMMANDs at the command prompt. You can also execute script files that contain the SQL statements.

solidDB Remote Control (solcon)
: solidDB Remote Control (**solcon**) is a console tool for administration; users with administrator rights can issue ADMIN COMMANDs at the command prompt or by executing a script file that contains the commands. With **solcon**, the ADMIN COMMANDs can be issued as part of the **solcon** startup command line.

  Because only users with administrator rights can access **solcon**, if only **solcon** is deployed at a production site, the administrators cannot accidentally execute SQL statements that could change the data.

Additionally, if you are using solidDB with the linked access methods, the solidDB Server Control API (SSC API) gives the application programmatic control over task execution. The SSC API functions are 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 more details on the SSC API, see *IBM solidDB Shared Memory Access and Linked Library Access User Guide*.

Configuration

Most solidDB server configuration settings are defined using configuration parameters. There are two `solid.ini` configuration files, one for the server and one for the ODBC client. Neither configuration file is obligatory. If there is no configuration file, the factory values are used. Also, all parameters do not need to be present in the `solid.ini` file. If a parameter is not present in the `solid.ini` file or if the value for a particular parameter is not set, the factory value is used.

You can set the configuration parameter values by editing the `solid.ini` configuration file manually or, in most cases, using ADMIN COMMANDs.

Some parameter settings can also be overridden per session or per transaction by using the SQL commands SET or SET TRANSACTION. You can also define the settings per connection with the ODBC connection attributes or JDBC connection properties. The precedence hierarchy is the following (from high precedence to low precedence):

- SET TRANSACTION: transaction-level settings
- SET: session-level settings
- ODBC connection attributes and JDBC connection properties
- Parameter settings specified by the value in the `solid.ini` configuration file
- Factory value for the parameter

Additionally, you can control some solidDB server operations with the following options:

- solidDB command-line options at startup
- environment variables
- ODBC and JDBC client connect string arguments

Related concepts:

2.3, “Creating a new database,” on page 13
solidDB databases are created at solidDB server startup.

3, “Configuring solidDB,” on page 37
The various solidDB configuration options help you to meet your environment, performance, and operation needs.

Related reference:

Appendix A, “Server-side configuration parameters,” on page 213
The server-side configuration parameters define various performance, memory and disk usage, and other characteristics of the solidDB server. Generally, the factory value settings offer the best performance and operability, but in some special cases modifying a parameter might improve performance.

Appendix B, “Client-side configuration parameters,” on page 267
The client-side configuration parameters define various characteristics for usage of the solidDB ODBC client and solidDB tools such as solidDB SQL Editor (**solsql**). The client-side parameters are stored in the client-side `solid.ini` configuration file and are read when the client starts.

Appendix C, “solidDB command-line options,” on page 271

Appendix D, “Environment variables,” on page 275

Appendix F, “solidDB ADMIN COMMAND syntax,” on page 359
This appendix describes the solidDB ADMIN COMMAND syntax. This command set is not part of ANSI SQL; it is an extension that is specific to the solidDB product.

2 Administering solidDB

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

- Performing basic solidDB operations, such as starting and stopping the server
- Backing up the server
- Creating checkpoints
- Compacting database files

Important: If you are using solidDB with shared memory access (SMA) or linked library access (LLA), there are some differences in administration from standard solidDB. Wherever necessary, references to information specific to the SMA or LLA configurations in the *IBM solidDB Shared Memory Access and Linked Library Access User Guide* are provided.

2.1 Automated and manual administration

The solidDB server is designed for continuous, unattended operation and ease of deployment. It requires minimal maintenance. Administrative operations, including backups, can be performed programmatically using **ADMIN COMMAND** SQL extensions which can run automatically or at the request by an administrator. When necessary, you can administer solidDB also manually.

The solidDB server provides the following command-line tools for administration:

solidDB SQL Editor (solsql)
: solidDB SQL Editor (**solsql**) is a console tool that you can use to issue SQL statements and solidDB ADMIN COMMANDs at the command prompt. You can also execute script files that contain the SQL statements.

solidDB Remote Control (solcon)
: solidDB Remote Control (**solcon**) is a console tool for administration; users with administrator rights can issue ADMIN COMMANDs at the command prompt or by executing a script file that contains the commands. With **solcon**, the ADMIN COMMANDs can be issued as part of the **solcon** startup command line.

  Because only users with administrator rights can access **solcon**, if only **solcon** is deployed at a production site, the administrators cannot accidentally execute SQL statements that could change the data.

Additionally, if you are using solidDB with the linked access methods, the solidDB Server Control API (SSC API) gives the application programmatic control over task execution. The SSC API functions are 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 more details on the SSC API, see *IBM solidDB Shared Memory Access and Linked Library Access User Guide*.

You can also schedule a subset of ADMIN COMMANDs to be executed using the **Srv.At** connection parameter. See 2.9, “Entering timed commands,” on page 34 for more details.

2.2 Starting and stopping server

2.2.1 Starting solidDB

You can start solidDB by issuing the command `solid [options]` at the command prompt, or in Windows environments, using the **Start** > **Programs** > **IBM solidDB** menu path.

Before you begin

To start solidDB, a valid license file must be located in your working directory or in the location specified with a SOLIDDIR environment variable.

About this task

Note: This section applies to standard solidDB only. If you are using solidDB with shared memory access (SMA) or linked library access (LLA), see the *IBM solidDB Shared Memory Access and Linked Library Access User Guide* for instructions on how to start a SMA or LLA server.

Procedure

To start the solidDB server:

Operating system	Procedure
Linux and UNIX	In the working directory, enter the command `solid [options]` at the command prompt. When you start the server for the first time, use the command-line option `-f` to force the server to run in the foreground: `solid -f`
Windows	• Click the icon labeled **Start IBM solidDB server** through the **Start** > **Programs** > **IBM solidDB** menu path. • In the working directory, enter the command `solid [options]` at the command prompt. • To start the server to run in the background, enter the command `start solid`.

options are

Option	Description	Examples
-c *directory*	Changes working directory	`solid -c /data/solid`
-f	Starts the server in foreground	
-m	Enables the monitoring facility for tracing SQL statements. For more details, see 5, "Monitoring solidDB," on page 91.	
-n *name*	Sets the server name	

Option	Description	Examples

-s install,*name,fullexepath* -c *directory*[,autostart]	The 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 a Windows service instance of the solidDB server	`solid -s"remove,SOLID"`
-s start	Specifies that solidDB starts 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"`
-U *username*	Specifies the username for the database that is being created. See also options **-x execute**, **- x executeandnoexit**, and **-x exit**.	
-P *password*	Specifies the password for the database that is being created. See also options **-x execute**, **- x executeandnoexit**, and **-x exit**.	
-p	Create a new database with externally authenticated database administrator	

Option	Description	Examples
-E	Encrypts the database. An encryption password is mandatory when **-E** is specified. The encryption password is needed to protect the symmetric encryption key which is stored in an unencrypted header page of the database file. Specify the encryption password using the **-x keypwdfile:***file_name* or **-S** *encryption_password* option.	`solid -C mycatalog -U admin -P admin123 -E -x keypwdfile:pwd.txt` `solid -C mycatalog -U admin -P admin123 -E -S admin456`
-S *encryption_password*	Specifies the database file encryption password	
-x assert:s	Disables emergency exit dialog	
-x autoconvert	Converts (migrates) the database format from a previous release version to the current release version and starts the server	
-x backupserver	Used only in HotStandby setups. Starts the server in a netcopy listening mode. A server in the netcopy listening mode accepts only netcopy operations from the Primary server.	
-C *catalog*	Specifies the database catalog name	
-x convert	Converts (migrates) database format to the current format used by solidDB and starts the server process	

-x decrypt -S *password*	Decrypts the database	`solid -x decrypt -S dba`
		
		`solid -x decrypt -x keypwdfile:pwd.txt`

-x disableallmessageboxes	Hides all message windows	
-x errormsgnostop	Does not wait for user actions on error dialogs	

-x execute: *file_name*	Prompts for the user name and password of the database administrator, creates a new database, executes SQL statements from a file, and exits.
	
	You can also use the options **-U** and **-P** to provide the DBA 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`

Option	Description	Examples

-x executeandnoexit: *file_name*	Prompts for the user name and password of the database administrator, creates a new database, executes SQL statements from a file, but does not exit.
	
	You can also use the options **-U** and **-P** to provide the DBA 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`

-x exit	Prompts for the user name and password of the database administrator, creates a new database, and exits. You can also use the options **-U** and **-P** to provide the DBA user name and password.	`solid.exe -x exit` `solid.exe -x exit -Udba -Pdba`
-x forcerecovery	Performs a forced roll-forward recovery	
-x hide	Hides the server icon	
-x ignoreerrors	Ignores index errors	
-x ignorecrashed	Ignores log files and reverts to checkpoint	
-x inifile:*file_name*	Specifies the configuration file name, instead of using the default `solid.ini` file in the working directory	
-x infodbfreefactor	Informs about unused pages The server exits after performing the task. See also:**-x reorganize**.	
-x keypwdfile: *file_name*	Reads the database encryption password from a file, instead of command line argument. This way the password cannot be seen by running the UNIX command **ps**.	
-x listen:*network_name*	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). • 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	Ignores log files during recovery	
-x pathprefix:*directory*	Uses files in the specified directory	

Option	Description	Examples
-x pwdfile: *file_name*	Reads the password from a file instead of command line argument. This way the password cannot 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.	
-x testintegrity	Performs a full database integrity test and exits	
-x testblocks	Checks the disk block integrity and produces a report in a `ssdebug.out` file. The server exits after performing the task.	
-x testindex*[:size]*	Tests database index and exits The optional *[:size]* parameter outputs index size.	
-x version	Displays the server version and exits	
-?	Help = Usage	
-h	Help = Usage	

Results

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

Related tasks:

2.5, “Running solidDB server as a Windows service,” on page 21
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.

Related reference:

Appendix C, “solidDB command-line options,” on page 271

Related information:

“FileSpec_[1...n] parameter” on page 48
The **Indexfile.FileSpec** parameter describes the location and the maximum size of an index file (database file).

Modifying Windows shortcuts for solidDB server and solidDB SQL Editor (solsql)

By default the **Start IBM solidDB server** and **solsql SQL Editor** icons in the **Start** > **Programs** > **IBM solidDB** menu path start programs in the `eval_kit\standalone` directory. To change the default settings, modify **Properties** of the shortcuts.

Modifying Start IBM solidDB server shortcut

By modifying the **Properties** of the **Start IBM solidDB server** shortcut you can specify the working directory, login data and system catalog name, and additional command-line options used when starting solidDB.

1. Right-click on the **Start IBM solidDB server** icon.
2. Select **Properties** and then the **Shortcut** tab.
3. To change the login data and catalog name (or other startup options), modify the command-line options given in the **Target** field:
   - **-C** — system catalog name
   - **-P** — password
   - **-U** — username

   For example:

   ```
   "C:\Program Files\IBM\solidDB\solidDB7.0\bin\solid.exe" -C mycatalog -P mypassword -U myname
   ```

   See section Appendix C, “solidDB command-line options,” on page 271 for a list of available solidDB startup options.
4. To change the working directory, modify the directory path in the **Start in** field.

   For example:

   "C:\Program Files\IBM\solidDB\solidDB7.0\eval_kit\mytest\"

Modifying solsql SQL Editor shortcut

By modifying **Properties** of the **solsql SQL Editor** shortcut you can specify the connection information and the login data for the solidDB server to which solidDB SQL Editor (**solsql**) connects to.

1. Right-click on the **solsql SQL Editor** icon.
2. Select **Properties** and then the **Shortcut** tab.
3. To change the connection information and login data, modify the server name, username, and password given in the **Target** field.

   For example:

   ```
   "C:\Program Files\IBM\solidDB\solidDB7.0\bin\solsql.exe" "tcp 2315" myname mypassword
   ```

   You can also specify startup options in the **Target** field. See section 7.2.1, “Starting solidDB SQL Editor (**solsql**),” on page 134 for a list of available **solsql** startup options.

2.2.2 Closing a database (preventing new connections)

Closing the database means that new connections to the database are not allowed. Closing the database does not affect existing user connections.

About this task

You can close the database 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.

Procedure

To close the database, issue the following command:

```
ADMIN COMMAND 'close';
```

Results

After closing the database, only connections from solidDB Remote Control (**solcon**) are accepted.

Any new connections fail with the error `solidDB Server Error 14506: The server is closed, no new connections allowed`.

What to do next

To view all existing connections, issue the following command:

```
ADMIN COMMAND 'userlist'
```

To disconnect specified users from the database, issue the following command:

```
ADMIN COMMAND 'throwout {username | userid}
```

To revert the effect of the **close** command, issue the following command:

```
ADMIN COMMAND 'open'
```

Related tasks:

2.2.3, “Shutting down solidDB”

2.2.3 Shutting down solidDB

About this task

This section applies to standard solidDB only. If you are using solidDB with shared memory access (SMA) or linked library access (LLA), see the *IBM solidDB Shared Memory Access and Linked Library Access User Guide* for instructions on how to stop the solidDB SMA or LLA server.

Procedure

You can shut down the solidDB in the following ways:

- To shut down solidDB programmatically from an application, for example, using solidDB Remote Control (**solcon**) or solidDB SQL Editor (**solsql**):
  1. Prevent new connections to solidDB by issuing the following command:

     ```
     ADMIN COMMAND 'close'
     ```

     Tip: To revert the effect of the close command, issue the **ADMIN COMMAND 'open'** command.
  2. Exit all users of solidDB (except the current connection) by issuing the following command:

     ```
     ADMIN COMMAND 'throwout all'
     ```

     The **throwout all** command does not wait for open transactions to finish; it rolls back all open transactions.
  1. Stop solidDB by issuing the following command:

     ```
     ADMIN COMMAND 'shutdown'
     ```
- To shut down solidDB using a single command, issue the following command:

```
ADMIN COMMAND 'shutdown force'
```

The **shutdown force** command performs the same operations as **close**, **throwout all**, and **shutdown**.

- In Windows environments, right-click the server icon and select **Close**.
- In Windows environments, if you are running solidDB as a service, you can use the commands**net stop** or **sc stop** through the Windows system services.

Results

All of the above-mentioned shutdown mechanisms start the same routine; all buffered data is written to the database file, cache memory is freed, and the server is shut down. The server shutdown can take a while since the server must write all buffered data from main memory to the disk.

Related tasks:

2.5, "Running solidDB server as a Windows service," on page 21
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.

2.3 Creating a new database

solidDB databases are created at solidDB server startup.

When solidDB is started, it checks if a database exists. If a database does not exist, solidDB prompts you to create a database.

- In Windows environments, a dialog window prompts you for the username and password of the database administrator, and a name for the default (system) catalog.
- In Linux and UNIX environments, 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 username and password of the database administrator, and a name for the default (system) catalog.

After accepting the username and password of the database administrator, solidDB creates the database.

By default the database is created as one file (`solid.db`) in the solidDB working directory.

An empty database that contains only the system tables and views uses approximately 4 MB of disk space. The time it takes to create the database depends on the hardware platform you are using. If you have a small database (less than 4 MB) and want to keep the disk space less than 4 MB, set the value of the **IndexFile.ExtendIncrement** parameter to less than 500 (default). This parameter and other parameters are discussed in Appendix A, "Server-side configuration parameters," on page 213.

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

2.3.1 Usernames, passwords, and system catalog names

The database system administrator account is created when the solidDB database is created; the creator of the database has the SYS_ADMIN_ROLE user role. The system catalog name is also created when the database is created and it cannot be changed later.

By default, the solidDB server offers a traditional authentication mechanism in which a user has to provide a valid user ID and password combination to connect to a database. Alternatively, you can configure solidDB to use an operating-system-based external authentication mechanism.

Important:

- You must remember your username and password to be able to connect to solidDB. There are no default user names. The administrator username you enter when creating the database is the only username available for connecting to the new database for the first time. If you cannot connect to solidDB because you have forgotten your system administrator user name or password, contact IBM Software Support.
- Lowercase characters in user names, passwords, and system catalog names are converted to uppercase.
- If you try to log in four times with an incorrect username or password, the system blocks your IP address for a maximum of 60 seconds. This feature cannot be configured or switched off.

Username

- Minimum length: 2 characters.
- Maximum length: 80 characters
- The username must begin with a letter or an underscore. Use lowercase letters from a to z, uppercase letters from A to Z, the underscore character "_", and numbers from 0 to 9.

The database system administrators username cannot be changed with the ALTER USER command. See *Changing DBA username and password* in the *IBM solidDB SQL Guide*.

Password

- Minimum length: 3 characters.
- Maximum length: 80 characters
- The password can begin with any letter, underscore, or number. Use lowercase letters from a to z, uppercase letters from A to Z, the underscore character "_", and numbers from 0 to 9.
- You cannot use the double quotation mark (") in the password. The use of apostrophe ('), semicolon (;), or space (' ') is discouraged, because some tools might not accept these characters in the password.
- If you plan to use solidDB Remote Control (**solcon**), do not create passwords with non-ASCII characters, because **solcon** does not perform UTF-8 translation for any input.
- You can also enter the password from a file. For more information, see 7.6, “Entering password from a file,” on page 158.
- The solidDB passwords do not expire. If you want set up user accounts with expiring passwords, use the operating-system-based mechanism for authentication.

System catalog

- Minimum length: 1 character.
- Maximum length: 39 characters
- The system catalog name must not contain spaces.

The solidDB syntax for database object hierarchy is the following:

```
catalog_name.schema_name.database_object
```

The default schema name is the username.

If you do not specify the catalog and schema name, the server uses the system catalog and the username of the object creator to determine which object to use.

For details on solidDB catalogs and schemas, see section *Managing database objects* in *IBM solidDB SQL Guide*.

Related concepts:

4.1, "Authentication," on page 57
Authentication is the mechanism of verifying the identity of a user or an application. By default, the solidDB server offers a traditional authentication mechanism in which a user has to provide a valid user ID and password combination to connect to a database. Alternatively, you can configure solidDB to use an operating-system-based external authentication mechanism.

4.1.2, "Operating-system-based external authentication," on page 58
Instead of the internal solidDB authentication mechanism, the user can be authenticated by services provided by operating system.

2.3.2 Unicode and partial Unicode database modes

Starting from version 6.5, the solidDB databases can be created in two modes: *Unicode* mode or partial Unicode mode (default). This database mode is based on the encoding of character data types (CHAR, VARCHAR, and so on) in the solidDB server. Wide character data types (WCHAR, WVARCHAR, and so on) are Unicode encoded in both modes.

- Unicode mode

  In the Unicode mode, the internal representation for character data types is UTF-8.

  The internal representation for wide character data types is UTF-16.

- partial Unicode mode

  In the partial Unicode mode, the internal representation for character data types uses no particular encoding; instead, the data is stored in byte strings with the assumption that user applications are aware of this and handle the conversion as necessary.

  The internal representation for wide character data types is UTF-16.

  The databases created with solidDB version 6.3 or earlier are of the partial Unicode type.

  Important: The default database mode in 6.5 is partial Unicode.

Creating Unicode databases

The solidDB database mode is controlled with the parameter **`General.InternalCharEncoding`**.

- Unicode mode: **`General.InternalCharEncoding=UTF8`**

When the **InternalCharEncoding** is set to UTF8, the internal representation for character data types is UTF-8. Both character data types and wide character data types are converted between the solidDB server and the application.

- partial Unicode mode: **General.InternalCharEncoding=Raw**

  When the **InternalCharEncoding** is set to Raw, the internal representation for character data types uses no particular encoding; instead, the data is stored in byte strings with the assumption that user applications are aware of this and handled the conversion as necessary. Wide character data types are converted between the solidDB server and the application.

  The databases created with solidDB version 6.3 or earlier are of the partial Unicode type.

Important: The database mode must be defined when the database is created and it cannot be changed later.

If the database already exists in either mode and the database mode contradicts the value of the parameter, the server startup fails with the following error message in the solerr.out:

```
Parameter General.InternalCharEncoding contradicts the existing database mode
```

2.3.3 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 might 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 3. solidDB default files

File	Factory value location	How to modify
license file (solid.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.

Table 3. solidDB default files (continued)

File	Factory value location	How to modify
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 want to run the installer on your production environment node, install solidDB on a separate node and copy the executable programs, libraries, and drivers manually to your production node, as applicable for your setup.
- To prevent loss of data in a disk failure, store the database files and transaction log files on different physical drives. Using different physical drives also provides 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.

  Using local disks is especially important with a solidDB HotStandby setup. The HotStandby configurations are targeted for environments with shared nothing architecture. Such architecture 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 concepts:

5.1, “Viewing error messages and log files,” on page 92
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'`.

Related information:

3.1.1, “Configuration files and parameter settings,” on page 38
There are two different `solid.ini` configuration files, one for the server and one for the ODBC client. Neither configuration file is obligatory. If there is no configuration file, the factory values are used.

“Managing database files and caching (IndexFile section)” on page 47

2.7, “Performing backup and recovery,” on page 24

2.6, “Running several servers on one computer,” on page 23

2.3.4 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 you customize and optimize the solidDB database server. For example, the **`IndexFile.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. Similarly, the **`IndexFile.BlockSize`** parameter specifies the block size for the database. The block size affects performance and also limits the maximum record size.

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 213. For more details about setting parameters, see 3, "Configuring solidDB," on page 37.

2.3.5 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 the **`Indexfile.BlockSize`** value before creating a database. If you have an existing database, remember 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). You can also use the unit symbols of K for KB and M for MB.

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

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

- Name and location of the database files – the default file name is `solid.db` and the default location is 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, the maximum is 64 TB - 1.

You can also use the **`Indexfile.FileSpec`** parameter to divide the database file into multiple files and onto multiple disks. Multiple database files are useful if you create a large physical database.

For details on configuring the database file locations and sizes with the **`Indexfile.FileSpec`** parameter, read "Managing database files and caching (IndexFile section)" on page 47.

2.3.6 Defining database objects

The solidDB database objects include catalogs, schemas, tables, views, indexes, stored procedures, triggers, and sequences. By default, database object names are qualified with the user name of the object creator and a system catalog name. You can also specify that database objects are qualified by a schema name.

You can use catalogs and schemas to organize your data. The use of schemas in solidDB conforms to the SQL standard, but the use of catalogs is an extension to the SQL standard.

Catalogs and schemas allow you to group database objects (tables, sequences, and so on) in a hierarchical way. Typically you put related items into the same group. For example, all the tables related to the accounting system might be in one group (for example, a catalog), while all the tables related to the human resources system might be in another group. You can also group database objects by user. For example, all of the tables used by Jane Smith might be in a single schema.

Catalogs are the highest (broadest) level of the hierarchy. Schema names are the mid-level. Specific database objects, such as tables, are the lowest (narrowest) level of the hierarchy. Thus, a single catalog may contain multiple schemas, and each of those schemas may contain multiple tables.

Object names must be unique within a group, but do not have to be unique across groups. For example, the schemas named after the users Jane Smith and Robin Trower might each have a table named BILLS. These two tables have nothing to do with each other. They can have different structures and different data, even though they have the same name. Similarly, the catalog ACCOUNTING_CATALOG and the catalog HUMAN_RESOURCES_CATALOG might each have a schema named DAVID_JONES. Those schemas are unrelated to each other, even though they have the same name.

If you want to specify a particular table and the table name is not unique in the database, you can identify it by specifying the catalog, schema, and table name, for example, `ACCOUNTING_CATALOG.DAVID_JONES.BILLS`.

If you do not specify the complete name (that is, if you omit the schema, or the schema and the catalog), the server uses the current/default catalog and schema name to determine which table to use.

In general, a catalog can be thought of as a logical database. A schema typically corresponds to a user.

Size limitations on database objects

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.

You can set the maximum size of LONG VARCHAR (or CLOB) columns that are used in string functions by using the **`Sql.MaxBlobExpressionSize`** parameter. By default, the size is 1 MB (1 megabyte).

For efficiency, the solidDB server 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, the server automatically detects the BLOBs and stores the objects to a special file area that has optimized block sizes for large files. Administrative actions are not required. For more information, see section *BLOBs and CLOBs* in the *IBM solidDB SQL Guide*.

2.4 Connecting to solidDB with solidDB tools (`solsql` and `solcon`)

After starting solidDB, you can connect to the server from your workstation using, for example, the solidDB data management tools, solidDB SQL Editor (**`solsql`**) or solidDB Remote Control (**`solcon`**).

Note: This section applies to standard solidDB only. If you are using solidDB with shared memory access (SMA) or linked library access (LLA), see the *IBM solidDB Shared Memory Access and Linked Library Access User Guide* for instructions on how to connect to a SMA or LLA server.

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 tools and give the network name of the server as a command-line parameter:

Table 4. Connecting to solidDB

Tool	Command

solidDB Remote Control (**`solcon`**)	`solcon "`*`networkname`*`" [`*`userid`* `[`*`password`*`]]`
	
	For example:
	`solcon "tcp hobbes 1313"`
	
	If you do not specify the DBA user name and password on the command line, **`solcon`** prompts you to enter them.
	Important: You must have administrator rights (SYS_ADMIN_ROLE) to use **`solcon`**.

solidDB SQL Editor (**`solsql`**)	`solsql "`*`networkname`*`" [`*`userid`* `[`*`password`*`]]`
	
	For example:
	`solsql "tcp hobbes 1313"`
	
	If you do not specify the DBA user name and password on the command line, **`solsql`** prompts you to enter them.

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

Related concepts:

6, “Managing network connections,” on page 117
Applications can connect to the solidDB server using network drivers or by linking to the server directly. The solidDB product supports multiple network protocols and connection types simultaneously.

Related information:

7, “Using solidDB data management tools,” on page 131
The solidDB product includes a set of data management tools which are

command-line utilities for performing various database tasks.

2.5 Running solidDB server 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.

2.5.1 Starting solidDB server as a service for the first time

The first time you want to run the solidDB server 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. If 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,<service_name>,<fullexepath> -c<working directory>[,autostart] [<option>]"
   ```

   where

   `<service_name>` is the name of the service

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

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

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

   Note:

   Regardless of the `autostart` parameter, the service is not started automatically at the time of installation. For the first time, the service has to be started manually in the Windows Services dialog or command prompt.

   `<option>` can be one of the Appendix C, “solidDB command-line options,” on page 271. For example, when using an encrypted database, the encryption password must be provided with the `-S<password>` option. **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 runs automatically as a service.

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

Example 3

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`. The solidDB database is encrypted; the encryption password is `abcd`.

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

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 GUI-based interactions.

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 <service_name>
     ```

     or

     ```
     net start <service_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.

2.5.2 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 <service_name>
    ```

    or

    ```
    net start <service_name>
    ```

– issue the following command to stop the service:

```
sc stop <service_name>
```

or

```
net stop <service_name>
```

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

2.5.3 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 <service_name>
     ```

     or

     ```
     net stop <service_name>
     ```

   where <service_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"
   ```

2.6 Running several servers on one computer

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

solidDB uses a concept of a working directory. Typically the working directory contains files related to running a particular solidDB instance:

- license file
- `solid.ini` configuration file
- database files
- transaction log files
- message and trace files

If you want to run several servers concurrently on one computer, you have to set up separate working directories for each solidDB instance.

To run several servers on one computer:

- Start each solidDB server process in its working directory,

  or
- Use the command-line option **-c** *directory_name* to change the working directory.

To avoid network conflicts, use different network listen names for each server in the `solid.ini` configuration files.

Example:

To start two solidDB server instances:

1. Create two working directories. For example:
   - `C:\solid1`
   - `C:\solid2`
2. Copy the license file into both directories.
3. In each working directory create a `solid.ini` configuration file, specifying different listen names.

   For example:

   solid1:

   ```
   [Com]
   Listen=tcpip 2315
   ```

   solid2:

   ```
   [Com]
   Listen=tcpip 2316
   ```
4. In the solidDB installation root directory:
   a. Start the first solidDB server instance with the following command:

      ```
      bin\solid -c C:\solid1
      ```

   b. Start the second solidDB server instance with the following command:

      ```
      bin\solid -c C:\solid2
      ```

2.7 Performing backup and recovery

Backups are made to secure the information stored in your database files. If your database files 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, back up the master database (and possibly the replica databases) regularly.

solidDB 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 local backup except that the backup database is sent over the network to Network Backup Server.

The following sections describe how to back up your solidDB databases and recover from system failure.

For guidelines for backing up and restoring the master and replica databases, see the *IBM solidDB Advanced Replication User Guide*.

2.7.1 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 5. 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 with the **`General.BackupDirectory`** directory. If the specified backup directory does not exist, solidDB returns error `Database Error 10030: Backup directory` *`directory name`* `does not exist`.

Important: If two databases are copied to the same directory, the earlier is overwritten by the latter. The *backup dir* must be different at least for each database. Moreover, although database files can be stored to different directories and partitions at the source server, they all are copied to the same backup directory. Therefore, identically named database files will conflict in the backup directory. As a consequence, only the most recent backup file among the identically named file has a backup copy in the backup directory.

2.7.2 Making backups over network

You can send a backup database over the network to any host running a solidDB server. A server playing the role of the backup receiver is called a *NetBackup Server*. By default, any solidDB server instance can acts as the NetBackup Server.

By default, the netbackup operation copies the database to one flat file in the NetBackupDirectory, even if the logical database consists of multiple files. Instead of flattening the structure to one file, you can define multiple files to which the source database files are mapped to during the netbackup operation. You map the source and target files in a `backup.ini` file that must be stored on the target server in the directory defined with the **`General.NetBackupDirectory`** parameter.

Making netbackup

Start a network backup (*netbackup*) with the following command:

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

where

- *options* can be:

Table 6. Options for the **netbackup** *command*

Option	Description
-s	Synchronized execution. The call returns either when the netbackup is completed or if there is an error.
-I	Executes a full database integrity check
-i	Executes a database index integrity check

- `DELETE_LOGS | KEEP_LOGS` defines whether backup logs are deleted or kept in the source server. Default is `DELETE_LOGS`.

Note:

- DELETE_LOGS is referred to as *Full backup*
- KEEP_LOGS is referred to as *Copy backup*. Using KEEP_LOGS corresponds to setting the **General.NetbackupDeleteLog** parameter to "no".

- connect *connect str* specifies the connection to the NetBackup Server. If *connect str* is omitted, it must be specified in the solid.ini configuration file. For the full connect string syntax, see "Format of the connect string" on page 52.
- dir *backup dir* defines the backup directory in the NetBackup Server. The path can be either absolute or relative to the **netbackup** root directory.

Important: If two databases are copied to the same directory, the earlier database copy is overwritten by the latter. The *backup dir* directory cannot 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 command ADMIN COMMAND 'status netbackup' is synonymous to ADMIN COMMAND 'status backup'; it reports on both local and network backups.
- The command ADMIN COMMAND 'netbackuplist' is synonymous to ADMIN COMMAND 'backuplist'; it 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 can 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, names, and sizes of the backup files stored into the file system of the NetBackup Server. The directories, names, and file sizes are specified in a backup.ini netbackup configuration file in the netbackup directory. The backup.ini file follows the syntax of [IndexFile] section in solid.ini configuration file. Therefore, in addition to the section name, the backup.ini file can 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 with a backup.ini file receives the incoming database as a whole, splits it into N separate parts, and stores the parts as files as specified in the backup.ini file.

Tip:

To retain the directory structure of the source server, 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 **IndexFile.FileSpec_[1...N]** specifications, creates similar directory structure, and stores backup files with their original properties to the NetBackup Server.

2.7.3 Configuring and automating backups

Both local and network backup settings can be configured in the `solid.ini` configuration file. If you want to automate backups, you can initiate the backup using the solidDB facility for entering timed commands.

The optional configuration settings for local and network backups can be set beforehand in the `solid.ini` configuration file, except for the synchronized execution, `-s` option. The following tables show the corresponding ADMIN COMMAND options and parameter settings.

Important: The options entered in ADMIN COMMAND override the corresponding parameter settings.

Table 7. **`ADMIN COMMAND 'backup'`** *and `solid.ini` parameter correspondence for local backups*

ADMIN COMMAND 'backup' option	Value	Parameter
`dir`	*backup dir*	**`General.BackupDirectory=`***backup dir* Default: `backup`

Corresponding ADMIN COMMAND options and configuration parameters for netbackup

Table 8. **`ADMIN COMMAND 'netbackup'`** *and `solid.ini` parameter correspondence for network backups*

ADMIN COMMAND 'netbackup' option	Value	Parameter
`connect`	*connect str*	**`General.NetBackupConnect=`***connect str* Default: no default
`dir`	*backup dir*	**`General.NetBackupDirectory`**=*backup dir* Default: no default
`netbackup DELETE_LOGS`		**`General.NetbackupDeleteLog`**=`yes` Default: yes
`netbackup KEEP_LOGS`		**`General.NetbackupDeleteLog`**=`no` Default: yes

Example: Setting local backup directory with the `General.BackupDirectory` parameter

1. Define the default backup directory with the **`General.BackupDirectory`** parameter.

   ```
   [General]
   BackupDirectory=weekly_backups
   ```

2. Start the backup by issuing the following command:

   ```
   ADMIN COMMAND 'backup'
   ```

Automating backups using timed commands

You can automate local backups by using the solidDB facility for entering timed commands.

To automate backups, use the **Srv.At** parameter to define the time and location of the backups.

```
[Srv]
At = [day] [HH:MM] backup [directory], [[day] [HH:MM backup [directory]]

day ::= sun | mon | tue | wed | thu | fri | sat
```

If you do not specify the location for the backup file with *directory*, the directory defined with the **General.BackupDirectory** parameter is used.

Automating daily backups using time commands

The following **Srv.At** setting starts a backup every day at 1:30 am. To ensure that at least a week old backup is available, the backups are stored in different directories each day.

```
[Srv]
At=sun 01:30 backup bckup_sun, mon 01:30 backup bckup_mon,
tue 01:30 backup bckup_tue,wed 01:30 backup bckup_wed,
thu 01:30 backup bckup_thu, fri 01:30 backup bckup_fri,
sat 01:30 backup bckup_sat
```

Related tasks:

2.9, “Entering timed commands,” on page 34
solidDB has a built-in timer which allows you to automate your administrative tasks. You can use timed commands to execute operating system commands, to create backups, checkpoints, and database status reports, to open and close databases, and to disconnect users and shut down servers.

Related reference:

A.4, “General section,” on page 217

Related information:

F.1, “ADMIN COMMAND,” on page 359

2.7.4 What happens during backup

Both local and network backups 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:

- Database files containing the checkpointed database itself
- Log files including changes made by those transactions that are active when the backup takes place
- `solmsg.out` database message file (provided for convenience in diagnosing problems — the message file is not required during a restore), and
- `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` license 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. The deletion is part of 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 must also ensure that the log files are not deleted at the end of checkpoint. To prevent deletion of log files, set the **General.CheckpointDeleteLog** parameter to `no`.

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 are 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 lose either your database files or backup files but not both. Similarly, if one file system fails, either the backup or the database files survives.

Network backup

Network backup (**netbackup**) is a facility for storing the whole database at some remote location. Netbackups are performed using 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 are 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 netbackup 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 if they are netbackup compatible. By principle, a newer version of the Netbackup Server can serve older versions of source servers. In other cases, the protocol version is checked and an incompatibility error is returned at the request for a netbackup. By principle, a newer version of the Netbackup Server can serve as a target server for an older version of source server.

2.7.5 Administering network backup server

Every solidDB database server since version 4.5 also acts as a Network Backup Server. However, you must specify the netbackup root directory using the **Srv.NetBackupRootDir** parameter.

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'**

   Cancels the backups in progress.
3. **ADMIN COMMAND 'shutdown"**

   Shuts down the server.

2.7.6 Monitoring and controlling backups

solidDB offers a set of ADMIN COMMANDs for monitoring and controlling backups.

The syntax is as follows:

```
ADMIN COMMAND 'command'
```

where *command* is any of the options presented in the following table.

Table 9. Backup and netbackup commands

Local backup command	Network backup command	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 ongoing backup process.

Querying the list of all completed backups and their success status

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

```
ADMIN COMMAND 'backuplist'
```

Canceling an active network backup operation

To cancel an active network backup operation, issue the following command:

```
ADMIN COMMAND 'abort netbackup'
```

2.7.7 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.

2.7.8 Troubleshooting backups

Backup media is out of disk space

Making a backup requires the same amount of disk space as the database being backed-up. Ensure that you have enough disk space in the backup storage device.

Invalid path for backup directory

The backup directory 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.

Local backup directory does not exist

If you specify a non-existent backup directory, the server prints 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.

Local backup directory is the same as the database directory

If the backup copies database files with their original names to the target directory, same source and target directory names leads to a file sharing conflict.

solidDB network backup server does not exist in the specified location

If you try to start a network backup without setting up solidDB network backup server properly, the netbackup fails.

Backup slows down the database

Backup can slow down the database if the backup uses same storage resources as the database. Slowdown can happen, for example, in the following cases:

- The backup write uses the same device controller as the database.
- The backup write uses the same physical storage device as the database.
- The operating system buffers large amounts of the backup data into memory.

2.7.9 Restoring backups

You can restore the database to the state it was in when the backup was created. 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

You might need to take the following two preliminary steps before a database can be recovered from remote backup files.

1. If the `backup.ini` file 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`, and so on.
3. Copy the database files from the backup directory to the database file directory.
4. Start solidDB.

Recovery is not performed 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 uses the log files automatically 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 startup. No administrative procedures are required to start the recovery.

2.7.10 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 incomplete blocks at each commit until it becomes full. It can 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 by setting the **Logging.LogEnabled** parameter to no.

Logging can be synchronous or asynchronous, depending on the transaction durability setting. For more information about transaction durability, see 8.1, "Logging and transaction durability," on page 163.

2.8 Creating checkpoints

A checkpoint updates the database file or files on disk. Specifically, the checkpoint copies pages from the memory cache of the database server to the database file on the disk drive. The server does the copy in a transactionally-consistent way; in other words, it copies only 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 has a consistent and valid (although not necessarily up-to-date) snapshot of the data.

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.

Checkpoints can be seen as 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 can also use part of the database file as swap space if the server cache overflows. In this situation, the server also writes to the database file.

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 8.7, "Tuning checkpoints," on page 176.

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

Note: There can be only 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 can affect the responsiveness of the server while the checkpoint is occurring.

Creating checkpoints manually

Before and after a database operation, you might want to create a checkpoint manually. To create a checkpoint manually, use the **ADMIN COMMAND 'makecp'** command.

You can also force a checkpoint using a timed command. See 2.9, “Entering timed commands” for more details.

2.9 Entering timed commands

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

Procedure

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

```
[Srv]
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 *day* is not specified, the command is executed daily.
The format used for time is HH:MM (24-hour format).
The following table contains a list of valid commands and their arguments.

Table 10. 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
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

Example

The following setting starts the following operations daily:

- Make checkpoint at 8:30 pm.
- Create a backup at 9:00 pm.
- Shutdown solidDB at 11 pm.

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

Related information:

F.1, “ADMIN COMMAND,” on page 359

2.10 Compacting database files (database reorganization)

Database reorganization returns unused space back to the file system. Reorganization is useful, for example, if your application causes short-term peaks in the database space usage, resulting in large allocated disk space. The database reorganization is started at solidDB startup with the command-line option `solid -x reorganize`.

About this task

When databases grow, solidDB server allocates new disk pages. 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. For example, you might want to use the reorganization feature to shrink the database size after you have deleted a large amount of data.

The solidDB database file compaction feature works in offline mode at the page level. Offline means that while a database file is being compacted, it 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; data is not moved among pages.

Important: The reorganization operation is not recoverable. If there is a failure during the reorganization operation, the reorganization or the database file cannot be recovered later. To avoid losing data, make a database backup before starting the reorganization.

Procedure

1. Make a backup copy of your database and log files.
2. Shut down the solidDB server.
3. View information about the database file size by starting the solidDB server with the following command:

   ```
   solid -x infodbfreefactor
   ```

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

   Example output:

   ```
   ------------------------------------------------------------
   2010-10-26 16:45:05
   IBM solidDB - Version 6.5.0.3 Build 2010-10-04 (Linux 2.6.18 AMD64 64bit MT)
   Infodbfreefactor option is activated.
   ------------------------------------------------------------
   Database file size = 152064 Kbytes
   Free blocks = 82128 Kbytes
   Log file size = 0 Kbytes
   Free space = 54.01%
   Block size = 16384 bytes
   ```

4. Start database reorganization by starting the solidDB server with the following command:

```
solid -x reorganize
```

The `-x reorganize` option invokes database reorganization. The operation moves pages to unused slots in the database file. When the page relocation is complete, the unused space is released back to the file system. The database 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 in the solidDB working directory.

5. Verify that the database size has been reduced by starting the server with the **`solid -x infodbfreefactor`** command.

3 Configuring solidDB

The various solidDB configuration options help you to meet your environment, performance, and operation needs.

Most solidDB server configuration settings are defined using configuration parameters. There are two `solid.ini` configuration files, one for the server and one for the ODBC client. Neither configuration file is obligatory. If there is no configuration file, the factory values are used. Also, all parameters do not need to be present in the `solid.ini` file. If a parameter is not present in the `solid.ini` file or if the value for a particular parameter is not set, the factory value is used.

Generally the factory values offer good performance and operability but in some cases modifying some parameter values can improve performance. You might also need to set configuration parameters to enable or disable certain functionality.

You can set the configuration parameter values by editing the `solid.ini` configuration file manually or, in most cases, using ADMIN COMMANDs.

Some parameter settings can also be overridden per session or per transaction by using the SQL commands SET or SET TRANSACTION. You can also define the settings per connection with the ODBC connection attributes or JDBC connection properties. The precedence hierarchy is the following (from high precedence to low precedence):

- SET TRANSACTION: transaction-level settings
- SET: session-level settings
- ODBC connection attributes and JDBC connection properties
- Parameter settings specified by the value in the `solid.ini` configuration file
- Factory value for the parameter

Additionally, you can control some solidDB server operations with the following options:

- solidDB command-line options at startup
- environment variables
- ODBC and JDBC client connect string arguments

Related reference:

Appendix A, “Server-side configuration parameters,” on page 213
The server-side configuration parameters define various performance, memory and disk usage, and other characteristics of the solidDB server. Generally, the factory value settings offer the best performance and operability, but in some special cases modifying a parameter might improve performance.

Appendix B, “Client-side configuration parameters,” on page 267
The client-side configuration parameters define various characteristics for usage of the solidDB ODBC client and solidDB tools such as solidDB SQL Editor (**`solsql`**). The client-side parameters are stored in the client-side `solid.ini` configuration file and are read when the client starts.

Appendix C, “solidDB command-line options,” on page 271

Appendix D, “Environment variables,” on page 275

3.1 Managing parameters

You can view and modify server-side configuration parameters using ADMIN COMMANDs or by editing the `solid.ini` configuration file. Client-side configuration parameters can only be viewed and modified using the `solid.ini` file.

3.1.1 Configuration files and parameter settings

There are two different `solid.ini` configuration files, one for the server and one for the ODBC client. Neither configuration file is obligatory. If there is no configuration file, the factory values are used.

- The server-side `solid.ini` is used as the main configuration file for the server.
- The client-side `solid.ini` file is used with the solidDB ODBC client (driver) and the solidDB data management tools (solidDB SQL Editor (**`solsql`**) and so on).

  Tip: If the solidDB server and the client are run on the same machine and use the same working directory, a single `solid.ini` configuration file can be both the server-side and the client-side configuration file. For example, the `solid.ini` configuration file in the `solidDB_installation_directory\eval_kit\standalone` directory contains both the server-side **`Com.Listen`** and the client-side **`Com.DataSources`** parameter settings.

Note: In solidDB documentation, `solid.ini` usually refers to the server-side `solid.ini` file.

When solidDB (or the ODBC client) 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 can also specify a different working directory by using the `-c` command-line option at solidDB startup.

If a value for a specific parameter is not set in the `solid.ini` file, solidDB uses the factory value for the parameter. The factory values can depend on the operating system you are using.

The configuration parameters are defined as *parameter name – value* pairs. The parameters are grouped according to section categories. Each section category starts with a section name inside square braces, for example:

```
[Com]
```

The `[Com]` section lists communication information. The section names are not case-sensitive. The section names `[COM]`, `[Com]`, and `[com]` are equivalent.

Tip: In documentation, parameters are typically referred to in the format **`section.parameter`**, for example, **`Logging.LogEnabled`**.

Example

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

Sample solid.ini files

The samples directory in the solidDB installation directory contains samples for different use cases. Each sample contains a solid.ini file with relevant settings for each use case; you can use the sample solid.ini files as a reference when configuring your environment.

Related concepts:

6.3.2, "Logical data source names," on page 123
The solidDB tools and client libraries support logical data source names. Logical data source names can be used for giving a database a descriptive name.

Related reference:

Appendix A, "Server-side configuration parameters," on page 213
The server-side configuration parameters define various performance, memory and disk usage, and other characteristics of the solidDB server. Generally, the factory value settings offer the best performance and operability, but in some special cases modifying a parameter might improve performance.

Appendix B, "Client-side configuration parameters," on page 267
The client-side configuration parameters define various characteristics for usage of the solidDB ODBC client and solidDB tools such as solidDB SQL Editor (**solsql**). The client-side parameters are stored in the client-side solid.ini configuration file and are read when the client starts.

3.1.2 Viewing and setting parameters with ADMIN COMMAND

You can change most server-side parameters with ADMIN COMMANDs without the need to restart the solidDB server. All parameters are accessible with the ADMIN COMMANDs even if they are not present in the solid.ini configuration file.

Viewing parameters

You can view the parameter settings by all parameters, all parameters in a section, or a single parameter at a time.

About this task

The syntax for viewing parameters is the following:

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

where:

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

Procedure

- To view all parameters, use the following command:

```
ADMIN COMMAND 'parameter';
RC TEXT
       -- ----
        0 Accelerator ImplicitStart Yes Yes Yes
        0 Accelerator ReturnListenErrors No No No
        0 Com Listen tcpip 2315, tcpip 2315, tcpip 1964
        0 Com MaxPhysMsgLen 8192 8192 8192
        0 Com RConnectLifetime 60 60 60
        0 Com RConnectPoolSize 10 10 10
        0 Com RConnectRPCTimeout 0 0 0
        0 Com ReadBufSize 2048 2048 2048
        0 Com SocketLinger Yes Yes Yes
```

```
         0 Com SocketLingerTime 0 0 0
         .
         .
         .
192 rows fetched.
```

- To view a single parameter, include the section name and parameter name in the command. For example:

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

- To view all parameters in a section, include the section name in the command. For example:

```
admin command 'parameter logging';
       RC TEXT
       -- ----
        0 Logging BlockSize 16384 16384 16384
        0 Logging DigitTemplateChar # # #
        0 Logging DurabilityLevel 1 1 1
        0 Logging FileFlush Yes Yes Yes
        0 Logging FileNameTemplate sol#####.log sol#####.log sol#####.log
        0 Logging LogDir logs logs
        0 Logging LogEnabled Yes Yes Yes
        0 Logging LogWriteMode 2 2 2
        0 Logging MinSplitSize 10485760 10485760 10485760
        0 Logging RelaxedMaxDelay 5000 5000 5000
        0 Logging SyncWrite No No No
11 rows fetched.
```

Results

The output show three values in the following order:

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

To show only the *current value*, use the `-r` option. For example:

```
admin command 'parameter -r logging';
       RC TEXT
       -- ----
        0 Logging BlockSize 16384
        0 Logging DigitTemplateChar #
        0 Logging DurabilityLevel 1
        0 Logging FileFlush Yes
        0 Logging FileNameTemplate sol#####.log
        0 Logging LogDir logs
        0 Logging LogEnabled Yes
        0 Logging LogWriteMode 2
        0 Logging MinSplitSize 10485760
        0 Logging RelaxedMaxDelay 5000
        0 Logging SyncWrite No
11 rows fetched.
```

Viewing the description of a specific parameter

You can view a detailed description of a specific parameter, which includes valid parameter types and access modes.

Note: Parameter support can vary between platforms.

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

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

Example

```
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 result set are:

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

Setting a parameter value

Most parameters can be changed with `ADMIN COMMAND 'parameter'`. Depending on the access mode of the parameter, the change might not apply immediately.

The syntax of the command is:

```
ADMIN COMMAND 'parameter param_name = value [temporary]'
```

- *param_name* and *value* follow the rules specified in 3.1.5, “Format of configuration parameter names and values,” on page 46.
  - *param_name* must include the section name and the parameter name, separated by a period.

    For example, to set the value of the **DurabilityLevel** parameter in the `[Logging]` section to '1', issue the command:

    ```
    ADMIN COMMAND 'parameter Logging.DurabilityLevel=1';
    ```
  - *value* must be a valid parameter value, or:

    If no value is specified, the parameter is set to the factory (or unset) value.

    If you assign a parameter value with an asterisk (*), the parameter is set to its factory value.
- You can provide blanks around the equal sign. For example:

  ```
  ADMIN COMMAND 'parameter com.trace = yes'
  ```

- When `temporary` is set, the changed value is not stored in the `solid.ini` file.
- When the value of a parameter is changed with an ADMIN command, the change might apply immediately or the next time that the server is started.
  - If a parameter value is written to the `solid.ini` file, it will take effect the next time that the server starts.
  - If the `temporary` option is used, the value affects the current behavior of the server, but does not affect the server when it restarts.
  - In some cases, the parameter change can be effective immediately. The parameter change is also written to the `solid.ini` file so that it also applies the next time that the server starts. The *access mode* of the parameter defines the persistence of the parameter change.

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

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

Related information:

3.1.4, "Access mode and persistence of parameter modifications," on page 45
The access mode of a parameter defines whether the parameter can be changed dynamically via an ADMIN COMMAND, and when the change takes effect.

3.1.3 Setting parameters through the `solid.ini` configuration file

When the solidDB server (or ODBC client) is started, it attempts to open the configuration file `solid.ini`. If the file does not exist, the factory values for the parameters are used. If the file exists, but a value for a particular parameter is not set in the `solid.ini` file, factory value for that parameter is used. 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.

You can specify a different directory to be used as the current working directory in the following ways:

- Use the `-c` solidDB command-line option.
- Set the SOLIDDIR environment variable to specify the location of the `solid.ini` file.

When searching for the `solid.ini` file, solidDB uses the following precedence (from high to low):

- location specified by the SOLIDDIR environment variable (if set)
- current working directory

Related reference:

Appendix C, "solidDB command-line options," on page 271

Rules for formatting the `solid.ini` file

The configuration file `solid.ini` is an ASCII file with line breaks. Comments are preceded with a semicolon (`;`).

```
[section_name]
param_name1=param_value
param_name2=param_value ;This is a comment

[section_name2]
param_name3=param_value
;This is a comment line (less than 79 characters)
```

Section names

The `solid.ini` configuration file is divided into sections. Each section contains a group of one or more related parameters.

Each section has a unique name. The name is delimited with square brackets. For example:

```
[SQL]
```

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.

Parameter names and values

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, the server uses 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.

Specifying duplicate parameter settings is not prohibited and does not result in an error message. The last occurrence of the parameter in the file takes the precedence.

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, the server uses the factory value.

Comments

The configuration file can contain comments; comments must begin with a semicolon (;). The comments can be put on separate lines or on the same line as a parameter.

```
; This line is a valid comment.
DurabilityLevel=2 ; This sentence is also a valid comment.
```

The maximum length of a line is 79 characters. If you create comments longer than 79 characters, the server splits the comments on separate lines using a backslash (\) at the end of the line but without adding a comment marker (;) on the new line. The server can handle the lines that have been split in this way; however, applications such as watchdogs might see the file as corrupted and thus fail.

Validation of entries

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 the entry is invalid, the server displays an error message in the `solmsg.out` file.

  If the server is running as a foreground process, the message is also displayed on the console.

The message is similar to one of the following:

- `Warning: Unrecognized entry in inifile: '<section>.<parameter>'.`

  You see this message if you have entries that fit the proper form, but which do not have the predefined section names and parameter names.

  For example, you can get this message with the following type of `solid.ini` entry:

  ```
  ; This line has a valid section name, but an invalid parameter name.
  [Logging]
  NoSuchParam=NoSuchValue

  ;This line 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.`

- `Warning: Illegal entry in inifile: <whole illegal line>`

  The server displays this message if a line could not be recognized as a section header, parameter name, comment, or blank line. You might see this message if you have entries that are not in the proper form.

  For example, you can get this message with the following type of `solid.ini` entry:

  ```
  ; This text was intended to be a comment
  but part of it is not preceded with a semicolon.
  ```

- `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.
- `Warning: Unregistered parameter <section>.<parameter> is used.`

  If this error occurs, it is a sign of a possible problem inside the server itself; report the error to IBM Software Support.

Important:

- The server does not necessarily display an error message if you use an invalid value for a parameter. The server simply uses 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.
- If you modify the `solid.ini` file and modify the parameters in the server by using an ADMIN COMMAND, the behavior is unpredictable. While the server is running, you can modify the `solid.ini` file or make changes to server values using the ADMIN COMMAND, but you cannot do both during the same run of the server.

Summary of `solid.ini` formatting rules

- Section name is in the format `[section_name]`
- The same section name can be used several times (not recommended).
- Each parameter is set in a separate line.
- The comment marker is the semicolon (`;`).
- Comments can follow other entries that are in the same line.
- The maximum length of a line is 79 characters.
- Entries in the files can be preceded with blanks.
- If the first non-blank character is the comment character, the whole line is ignored, that is, it is treated as a comment line.
- Lines that have no characters, or that have only blank characters, are ignored.
- The maximum length of a line is 78 characters.

Example

The following example shows a simple `solid.ini` file entry that contains a section heading, a parameter, and a comment:

```
[Logging]
; Use "relaxed logging", which improves performance but can
; risk losing the last few transactions during a failure.
DurabilityLevel=1

[Com]
...
```

3.1.4 Access mode and persistence of parameter modifications

The access mode of a parameter defines 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: the value can be changed via an ADMIN COMMAND and the change takes effect immediately.
- RW/Startup: the value can be changed via an ADMIN COMMAND and the change takes effect the next time that the server starts.
- RW/Create: the value can be changed via an ADMIN COMMAND and the change applies when a new database is created.

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

Saving parameters

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

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

If *ini_file_name* is not specified, the current `solid.ini` file is rewritten.

If *ini_file_name* is specified, a full configuration file is written to a new location. Specifying *ini_file_name* is is a convenient way to save configuration file checkpoints for later use.

Example: Read-only (RO) parameter `IndexFile.BlockSize`

The access mode of the **`IndexFile.BlockSize`** parameter is RO. The parameter is set when the database is created and cannot be modified afterward.

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

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.

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

3.1.5 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 can use any of the following: m, M, MB, mb, Mb, or mB. Some units (for example, 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 used in an ADMIN COMMAND, the parameter name must be the full parameter name, including the section name, for example, **Logging.DurabilityLevel**. When used in the solid.ini file, the parameter cannot include the section name, since the parameter must already be listed under the appropriate section header.

  value_literal is the value to be assigned to the parameter. The value is usually a literal, such as the number 12, or a string, such as "tcp MyServer2 1315". If you specify no value, the parameter is set to its startup value. If you assign a parameter value with an asterisk (*), the parameter is set to its factory value.

  Note: String literals normally need to be in double quotation marks 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
```

3.1.6 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 starts listening to one or more protocols with network names that distinguish it in the network. A client application uses a similar network name (connect string) to specify which protocol to use and which server to connect to.

The network name is defined with the **Listen** parameter in the [Com] section, for example:

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

The default value is operating system dependent. See 6, “Managing network connections,” on page 117 for details on the parameter format.

Managing database files and caching (IndexFile section)

In solidDB, data and indexes are stored in the same file or files. 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 or files used to store the database. The [IndexFile] section of solid.ini also controls the caching-related parameters.

FileSpec_[1...n] parameter:

The **Indexfile.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
- maximum file size
- device number (optional)

Example:

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

The default value for the **Indexfile.FileSpec** parameter is `solid.db 2147483647` (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. The maximum file size is (4G-1) *blocksize. With the default 16 KB block size, the maximum is 64 TB - 1.

The **Indexfile.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 **Indexfile.FileSpec** parameter identified by the number 2. The index file is written to the second file if it grows over the maximum value of the first **Indexfile.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). The 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 operating system.

Although the database files reside in different directories, the file names must be unique. In the 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 can use.

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

You might 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, the multithreaded solidDB can assign a separate disk I/O thread for each device. This way the server can perform database file I/O in a parallel manner.

The optional `device number` that you can specify for each data file helps the server optimize its performance. 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 other 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, on a Windows system that has two physical disk drives, the first physical disk drive is typically `C:`. A second physical disk drive could be partitioned into two logical disk drives, `D:` and `E:`. If one data file is put on `C:`, one on `D:`, and one on `E:`, 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 `E:`) are arbitrary and meaningless.

If your database has reached the maximum size specified by the **`FileSpec`** parameter, you need to increase the maximum file size limit or divide the database into multiple files.

Important: Do not attempt to use the **`FileSpec`** parameter to decrease the size of a database; you risk losing existing data and corrupting the database.

Related concepts:

9.1.5, “Troubleshooting database file size (file write fails),” on page 199
If your database has reached the maximum size specified by the **`IndexFile.FileSpec`** parameter, you need to increase the maximum file size limit or divide the database into multiple files.

CacheSize parameter:

The **`IndexFile.CacheSize`** parameter defines the amount of main memory that is used to maintain the shared buffer pool of a disk database. This buffer pool is called the database cache.

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. The absolute minimum size is 512 kilobytes.

Although the solidDB server is able to run with a small cache size, a larger cache size generally speeds up the server.

For a pure in-memory database (M-tables only), the cache size is mostly irrelevant, as long as it is not less than 8 MB.

Example

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

The size unit is bytes. You can also specify the amount of space in units of megabytes, for example, "10M" for 10 megabytes.

Related information:

"Defining database cache size" on page 169

Specifying default table storage type (General section)

By default, new tables are created as in-memory tables (M-tables). You can set the default table type with the **General.DefaultStoreIsMemory** parameter.

You can override the value set with **General.DefaultStoreIsMemory** by using the STORE clause in the CREATE TABLE statement.

For example:

```
CREATE TABLE employees (name CHAR(20)) STORE MEMORY;
CREATE TABLE ... STORE DISK;
ALTER TABLE network_addresses SET STORE MEMORY;
```

Specifying 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 because all the database files of one database are copied to the same directory. The backup directory can be set to any existing directory except the solidDB database file directory, the log file directory, or the working directory.

The name and location for your backup directory is defined with the **BackupDirectory** parameter in the `[General]` section.

The default location is a directory relative to your solidDB working directory.

For example:

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

With the above value 'backup', the backup is written to a directory that is a subdirectory of the solidDB directory.

The backup directory entered must be a valid path name in the 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 does not exist, it is created (write rights needed).

Source-side parameter: The parameter **General.NetBackupDirectory** in the source server sets the remote directory for use of Network Backup. The value of the parameter is either absolute or relative to the root directory of the NetBackup Server.

Netbackup Server parameter (Srv.NetBackupRootDir): The parameter **Srv.NetBackupRootDir** in the NetBackup Server sets the root directory for all netbackup operations in case the netbackup directory defined with the **General.NetBackupDirectory** parameter uses relative path expressions. The value of the **Srv.NetBackupRootDir** parameter can be absolute or relative to the working directory.

By default, the netbackup operation copies the database to one flat file in the NetBackupDirectory, even if the logical database consists of multiple files. Instead of flattening the structure to one file, you can define multiple files to which the source database files are mapped to during the netbackup operation. You map the source and target files in a `backup.ini` file that must be stored on the target server in the directory defined with the **`General.NetBackupDirectory`** parameter.

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

FileNameTemplate: The **`Logging.FileNameTemplate`** parameter defines a filename structure for the transaction log files. For example, the following setting instructs solidDB to create log files to directory `d:\logdir` and to name them sequentially starting from `sol00001.log`.

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

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

The filename can also be structured by using the **`Logging.FileNameTemplate`** parameter together with the **`Logging.LogDir`** parameter. The **`Logging.LogDir`** parameter defines the directory prefix of the filename and the **`Logging.FileNameTemplate`** parameter defines the actual filename. For more information, see A.7, “Logging section,” on page 234.

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 for more details.

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

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

The optimum number of threads depends on the number of processors the system has installed. Usually it is most efficient to have between 2 and 8 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

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*.

The SQL Info facility is turned on by setting the **`Info`** parameter in the `[SQL]` section to a non-zero value 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 on the tracing, the communication layer can log even the system-specific errors. System-specific errors can help in diagnosing the real problem in the network. For details, read "Network trace facility" on page 190. 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 about 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 about 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`. Be default, the `soltrace.out` is written to the current working directory of the server or client, depending on which end the tracing is started.

3.1.7 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 **Com.Connect** parameter defines the default connect string for a client to connect to when it communicates with a server. Because the client must use the same network name as the server is listening to, the value of the **Com.Connect** parameter on the client must match the value of the **Com.Listen** parameter on the server.

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 can used instead of the **Com.Connect** parameter.

Important: The `[HotStandby]` and `[Synchronizer]` sections in the `solid.ini` file also have **Connect** parameters. These parameters work independently from each other; however, they use the same format for the connect string.

Format of the connect string: A default connect string can be defined with the client-side **Com.Connect** configuration parameter. The connect string can also be supplied, for example, at connection time or when configuring data sources with an ODBC driver manager.

The same format of the connect string applies to the **Com.Connect** parameter as well as to the connect string used by solidDB tools or ODBC applications.

The format of a connect string is the following:

protocol_name [*options*] [*host_computer_name*] *server_name*

where

- *options* can be any combination of the following:

Table 11. Connect string options

Option	Description	Protocol
-4	Specifies that client connects using IPv4 protocol only.	TCP/IP
-6	Specifies that client connects using IPv6 protocol only. In Windows environments, this option is mandatory if IPv6 protocol is used.	TCP/IP
-i*source_address*	Specifies an explicit connecting socket source address for cases where the system default source IP address binding does not meet application needs. *source_address* can be an IP address or a host name.	TCP/IP
-z	Enables data compression for the connection **Important:** • Data compression is not available for HotStandby connections (**HotStandby.Connect**) and NetBackup connections (**ADMIN COMMAND 'netbackup'**). • Data compression for **netcopy** connections cannot be enabled with the **-z** option. Instead, use the **HotStandby.NetcopyRpcCompress=yes** parameter setting.	All
-c *milliseconds*	Specifies the login timeout (the default is operating-system-specific). A login request fails after the specified time has elapsed.	TCP/IP
-r *milliseconds*	Specifies the connection (or read) timeout. A network request fails when no response is received during the time specified. The value 0 (default) sets the timeout to infinite (operating system default timeout applies).	TCP/IP
-o*filename*	Turns on the Network trace facility and defines the name of the trace output file See *Network trace facility* in the *IBM solidDB Administrator Guide* for details.	All
-p*level*	Pings the server at the given level (0-5). Clients can always use the solidDB Ping facility at level 1 (0 is no operation/default). 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. See *Ping facility* in the *IBM solidDB Administrator Guide* for details.	All
-t	Turns on the Network trace facility See *Network trace facility* in the *IBM solidDB Administrator Guide* for details.	All

- *host_computer_name* is needed with TCP/IP and Named Pipes protocols, if the client and server are running on different machines.
- *server_name* depends on the communication protocol:
  - In TCP/IP protocol, server_name is a service port number, such as '2315'.
  - In other protocols, server_name is a name, such as 'soliddb' or 'chicago_office'.

  For details on the syntax in different communication protocols, see *Communication protocols* in the *IBM solidDB Administrator Guide*.

Note:

- The *protocol_name* and the server_name must match the ones that the server is using in its network listening name.
- If given at the connection time, the connect string must be enclosed in double quotation marks.
- All components of the connect string are case insensitive.

Examples

```
[Com]
Connect=tcp -z -c1000 1315
```

```
[Com]
Connect=nmpipe host22 SOLID
```

```
solsql "tcp localhost 1315"
```

```
solsql "tcp 192.168.255.1 1315"
```

```
rc = SQLConnect(hdbc, "upipe SOLID", (SWORD)SQL_NTS, "dba", 3, "dba", 3);
```

```
rc = SQLDriverConnect(hdbc,
                      (SQLHWND)NULL,
                      (SQLCHAR*)"DSN=tcp localhost 1964;UID=dba;PWD=dba",
                      38,
                      out_string,
                      255,
                      &out_length,
                      SQL_DRIVER_NOPROMPT);
```

Trace and TraceFile parameters: The client-side **Com.Trace** parameter controls whether solidDB collects trace information about network messages for the established network connection.

When **Com.Trace** is set to Yes, solidDB writes the trace log to the default trace file (soltrace.out) in the current working directory or to the file specified with the **Com.TraceFile** parameter.

3.2 Using solidDB command-line options

When starting solidDB, you can use command-line options, for example, to override certain parameter settings or invoke database operations such as database conversion.

About this task

- A full list of the available command-line options is available in section Appendix C, "solidDB command-line options," on page 271. You can also view the options with the command-line option -h or -?, For example:

  ```
  solid -h
  ```

- If the syntax of the command is incorrect, a list of the valid options is displayed.
- The command-line options are case-sensitive.

Procedure

At your operating system command prompt, use the following syntax:

```
solid [option] [option] [...]
```

Example

```
solid -Udba -Pdba -x listen:"tcp 2315" -E -Sadmin
```

The above command starts a solidDB server and encrypts an existing database where:

- `-U` = user name: `admin`
- `-P` = password: `admin123`
- `-x listen` = network listening name: `tcp 2315`
- `-E` = encrypts the database
- `-S` = encryption password: `admin`

3.3 Setting environment variables specific to solidDB

The environment variables specific to solidDB enable you to define default settings, for example, for the location of `solid.ini` file, license files, and trace files.

About this task

The solidDB-specific environment variables are listed in the following table.

Table 12. solidDB environment variables

Environment variable	Purpose	Example
SOLAPPINFO	Identifies applications running in the same computer and under the same username for the purposes of tracing and management SOLAPPINFO is set on the client node. The ADMIN COMMAND 'userlist' returns the value of SOLAPPINFO on the server side.The value of SOLAPPINFO must not contain blanks. **Tip:** In JDBC environments, the SOLAPPINFO can be set with the connection property solid_appinfo. Alternatively, the following Java™ command line may be used to pass the value of the environmental variable to the driver: `java -Dsolid_appinfo=%SOLAPPINFO% java_program_name`	`export SOLAPPINFO=testapp`
SOLIDDIR	Defines the default directory for `solid.ini` and license files	`export SOLIDDIR=/home/soliddb/settings/`
SOLSMASTART	Forces the start address space for the SMA server to the solidDB default The value depends on the operating system; see *SOLSMASTART default address spaces* in the *IBM solidDB Shared Memory Access and Linked Library Access User Guide* for more details.	`export SOLSMASTART=0x2c0000000000`
SOLTRACE	Turns on the Network trace facility, overriding the **`Com.Trace`** setting in the `solid.ini` file	`export SOLTRACE=yes`
SOLTRACEFILE	Defines the name and location of the file where trace information is output, overriding the **`Com.TraceFile`** setting in the `solid.ini` file Defining the SOLTRACEFILE environment variable automatically turns on the Network trace facility.	`export SOLTRACEFILE=/home/soliddb/settings/trace.out`

Procedure

- In Linux and UNIX environments, use following command:
  `export <environment_variable>=<value>`
- In Windows environments, use following command:
  `set <environment_variable>=<value>`

4 Security

The solidDB security methods help you keep your data secure by preventing unauthorized viewing and altering of data or database objects. solidDB also provides a secure-auditing facility of the database server.

4.1 Authentication

Authentication is the mechanism of verifying the identity of a user or an application. By default, the solidDB server offers a traditional authentication mechanism in which a user has to provide a valid user ID and password combination to connect to a database. Alternatively, you can configure solidDB to use an operating-system-based external authentication mechanism.

4.1.1 Default solidDB authentication

By default, the solidDB server offers a traditional (internal) authentication mechanism in which a user has to provide a valid user ID and password combination to connect to a database. By default, passwords are encrypted using the build-in DES algorithm.

Username

- Minimum length: 2 characters.
- Maximum length: 80 characters
- The username must begin with a letter or an underscore. Use lowercase letters from a to z, uppercase letters from A to Z, the underscore character "_", and numbers from 0 to 9.

The database system administrators username cannot be changed with the ALTER USER command. See *Changing DBA username and password* in the *IBM solidDB SQL Guide*.

Password

- Minimum length: 3 characters.
- Maximum length: 80 characters
- The password can begin with any letter, underscore, or number. Use lowercase letters from a to z, uppercase letters from A to Z, the underscore character "_", and numbers from 0 to 9.
- You cannot use the double quotation mark (") in the password. The use of apostrophe ('), semicolon (;), or space (' ') is discouraged, because some tools might not accept these characters in the password.
- If you plan to use solidDB Remote Control (**solcon**), do not create passwords with non-ASCII characters, because **solcon** does not perform UTF-8 translation for any input.
- You can also enter the password from a file. For more information, see 7.6, “Entering password from a file,” on page 158.
- The solidDB passwords do not expire. If you want set up user accounts with expiring passwords, use the operating-system-based mechanism for authentication.

Important:

- You must remember your username and password to be able to connect to solidDB. There are no default user names. The administrator username you enter when creating the database is the only username available for connecting to the new database for the first time. If you cannot connect to solidDB because you have forgotten your system administrator user name or password, contact IBM Software Support.
- Lowercase characters in user names, passwords, and system catalog names are converted to uppercase.
- If you try to log in four times with an incorrect username or password, the system blocks your IP address for a maximum of 60 seconds. This feature cannot be configured or switched off.

Encryption of passwords

By default, the passwords of internally authenticated users are encrypted using the build-in DES algorithm. The default encryption can be disabled by setting the **General.UseEncryption** parameter to no.

By default, the passwords of internally authenticated users are sent over the network connection in a scrambled format. However, you can use the IBM Global Security Kit (GSKit) to enable strong encryption when sending the password over the network connection. To enable strong encryption when sending passwords of internally authenticated users over a network connection, set the **General.GSKitLoginRequired** parameter to yes. If you set the **General.GSKitLoginRequired** parameter to yes, the GSKit must be installed and enabled on the client computer.

4.1.2 Operating-system-based external authentication

Instead of the internal solidDB authentication mechanism, the user can be authenticated by services provided by operating system.

The operating-system-based external authentication is supported on Linux, UNIX, and Windows environments. On Linux and UNIX systems, solidDB uses services provided by Pluggable Authentication Modules (PAM) API, implementing the X/Open Single Sign-On standard. On Windows systems, external authentication is implemented on top of Security Support Provider Interface (SSPI) API.

Additionally, to use external authentication, the IBM Global Security Kit (GSKit) must be enabled and accessible on both the server and client computers. GSKit enables use of private key/public key pair for the connect message, providing strong encryption when sending the password over the network connection.

Principles of operation

When external authentication is used, the user logs in to solidDB by providing authentication credentials that match the credentials of an operating system user account on the solidDB host computer.

To create an externally authenticated user account for the database administrator, you need to enable the external authentication when creating the database. For other users, you enable external authentication using SQL statements. The authentication of each user must be specified separately. Each externally authenticated solidDB user must have a corresponding operating system or domain level account on the machine where solidDB is running.

Additional security considerations

- If the user accounts are externally authenticated, the database and all of its backups must reside on encrypted or otherwise protected media; this is to ensure, for example, that malicious users cannot copy the database to another system and configure external authentication so that the login succeeds for any account.

Installing and configuring IBM Global Security Kit (GSKit) for external authentication

To use external authentication, the IBM Global Security Kit (GSKit) must be available on the solidDB server and client computers.

Note: When you enable the use of GSKit for external authentication, only the passwords are encrypted using GSKit. To encrypt the solidDB database and log files, you need to enable the encryption separately. For more details, see 4.2.2, "Encrypting database and log files," on page 78.

Installing and configuring IBM Global Security Kit (GSKit) for external authentication – server:

To use external authentication, you must install the IBM Global Security Kit (GSKit) on the solidDB server computer. If the solidDB server cannot access the GSKit library, the login data for an externally authenticated user cannot be verified. On the server side, the use of GSKit for encryption is controlled with the **`General.UseGSKit`** parameter.

About this task

The GSKit library is installed during solidDB installation. The solidDB installation program installs the GSKit libraries that the solidDB server and clients must be able to load. The GSKit installation includes also a set of auxiliary libraries.

The GSKit libraries for the most common platforms are shown in the table below:

Table 13. GSKit libraries and default installation locations

Platform	GSKit library	GSKit default installation location	Auxiliary library directories
Windows	`gsk8iccs.dll` `gsk8iccs_64.dll`	`<solidDB installation directory>\bin`	`<solidDB installation directory>\bin\C` `<solidDB installation directory>\bin\N`
Linux	`libgsk8iccs.so` `libgsk8iccs_64.so`	`<solidDB installation directory>/bin`	`<solidDB installation directory>/bin/C` `<solidDB installation directory>/bin/N`
Solaris	`libgsk8iccs_64.so`	`<solidDB installation directory>/bin`	`<solidDB installation directory>/bin/C` `<solidDB installation directory>/bin/N`

Table 13. GSKit libraries and default installation locations (continued)

Platform	GSKit library	GSKit default installation location	Auxiliary library directories
HP-UX	libgsk8iccs_64.so	<solidDB installation directory>/bin	<solidDB installation directory>/bin/C <solidDB installation directory>/bin/N
AIX®	libgsk8iccs_64.so	<solidDB installation directory>/bin	<solidDB installation directory>/bin/C <solidDB installation directory>/bin/N

Procedure

1. Ensure that the GSKit library and auxiliary libraries are available on the computer where solidDB server is running.
2. On the server computer:
   a. Set the **General.UseGSKit** parameter to yes.
   b. Set the **General.GSKitPath** parameter to point to the directory where the GSKit library is located.
   c. Optional: If you want that the passwords of any internally authenticated users are sent over a network connection using strong encryption, set the **General.GSKitLoginRequired** parameter to yes.

   For example:

```
[General]
UseGSKit=yes
GSKitPath=/home/IBM/solidDB/soliddb-7.0/bin/
GSKitLoginRequired=yes
```

What to do next

Install and configure IBM Global Security Kit (GSKit) for external authentication on the client computer. The configuration procedure is different depending on whether you use the solidDB JDBC client or the solidDB ODBC client or solidDB tools such as solidDB SQL Editor (**solsql**).

Related tasks:

"Installing and configuring IBM Global Security Kit (GSKit) for external authentication – ODBC clients and solidDB tools" on page 61
If you are using the solidDB ODBC Driver or solidDB data management tools (for example, solidDB SQL Editor (**solsql**)), to use external authentication, you must install the IBM Global Security Kit (GSKit) on the solidDB client computer. If the solidDB client cannot access the GSKit library, the login data for an externally authenticated user cannot be verified.

"Installing and configuring IBM Global Security Kit (GSKit) for external authentication – JDBC clients" on page 63
To use external authentication with JDBC, you enable the use of the IBM Global Security Kit (GSKit) using JDBC connection properties. You must also ensure that the solidDB JDBC Driver has access to the solidDB linked library access (LLA) and GSKit libraries. If the JDBC client cannot load the GSKit and LLA libraries, the login data for an externally authenticated user cannot be verified.

Installing and configuring IBM Global Security Kit (GSKit) for external authentication – ODBC clients and solidDB tools:

If you are using the solidDB ODBC Driver or solidDB data management tools (for example, solidDB SQL Editor (**solsql**)), to use external authentication, you must install the IBM Global Security Kit (GSKit) on the solidDB client computer. If the solidDB client cannot access the GSKit library, the login data for an externally authenticated user cannot be verified.

About this task

You can install the GSKit on the client computer in two ways:

- Use the solidDB installation program to install the solidDB package on the client computer. The GSKit library is installed during solidDB installation.
- Copy the GSKit library and auxiliary libraries to the client computer manually. For details, see the instructions below.

Procedure

1. If your client computer does not have a solidDB installation, copy the GSKit library and auxiliary libraries from the computer where you have installed solidDB server to the client computer. The GSKit library names and default installation locations are shown in the table below:

Table 14. GSKit libraries and default installation locations

Platform	GSKit library	GSKit default installation location	Auxiliary library directories
Windows	gsk8iccs.dll gsk8iccs_64.dll	<solidDB installation directory>\bin	<solidDB installation directory>\bin\C <solidDB installation directory>\bin\N
Linux	libgsk8iccs.so libgsk8iccs_64.so	<solidDB installation directory>/bin	<solidDB installation directory>/bin/C <solidDB installation directory>/bin/N
Solaris	libgsk8iccs_64.so	<solidDB installation directory>/bin	<solidDB installation directory>/bin/C <solidDB installation directory>/bin/N
HP-UX	libgsk8iccs_64.so	<solidDB installation directory>/bin	<solidDB installation directory>/bin/C <solidDB installation directory>/bin/N
AIX	libgsk8iccs_64.so	<solidDB installation directory>/bin	<solidDB installation directory>/bin/C <solidDB installation directory>/bin/N

2. Enable the use of GSKit for external authentication.
   - If your setup uses a client-side solid.ini file (or if you use solidDB tools):
     a. Set the client-side **Client.UseGSKit** parameter to yes.

b. Set the client-side **Client.GSKitPath** parameter to point to the directory where the GSKit library is located.

- If your setup does not use a client-side `solid.ini` file:
  a. Include the connection attribute USE_GSKIT_ENCRYPTION=YES in the ODBC connect string.
  b. Add the location of the GSKit library to the LD_LIBRARY_PATH or LIBPATH (Linux and UNIX) or PATH (Windows) environment variable.
     – In Linux and UNIX environments, use the following syntax:

       ```
       export LD_LIBRARY_PATH=<path to library>:$LD_LIBRARY_PATH
       ```

       or

       in AIX environments:

       ```
       export LIBPATH=<path to library>:$LIBPATH
       ```

     – In Windows environments, use the following syntax:

       ```
       set PATH=<path to library>=;%PATH%
       ```

       For example:

       ```
       set PATH="C:\Program Files\IBM\solidDB\solidDB7.0\bin";%PATH%
       ```

Example: Client-side `solid.ini` settings

Linux and UNIX operating systems:

```
[Client]
UseGSKit=yes
GSKitPath=/home/IBM/solidDB/soliddb-7.0/bin/
```

Windows operating systems:

```
[Client]
UseGSKit=yes
GSKitPath="C:\Program Files\IBM\solidDB\solidDB7.0\bin"
```

Tip: If the path contains a white space, enclose the path in double quotation marks.

Related concepts:

4.6.2, “Connection for an externally authenticated user fails at SQLAllocEnv,” on page 88

Related tasks:

“Creating externally authenticated accounts for database administrators” on page 70
The external authentication method for a database administrator account must be specified when creating a database. To create a new database with external authentication, use the solidDB startup option **-p** and omit the password.

“Creating externally authenticated accounts for users” on page 71
To enable the external authentication method for a user, use the CREATE USER or ALTER USER statements. You need to use the keyword EXTERNALLY and omit the password.

“Installing and configuring IBM Global Security Kit (GSKit) for external authentication – server” on page 59
To use external authentication, you must install the IBM Global Security Kit (GSKit) on the solidDB server computer. If the solidDB server cannot access the GSKit library, the login data for an externally authenticated user cannot be verified. On the server side, the use of GSKit for encryption is controlled with the **General.UseGSKit** parameter.

Installing and configuring IBM Global Security Kit (GSKit) for external authentication – JDBC clients:

To use external authentication with JDBC, you enable the use of the IBM Global Security Kit (GSKit) using JDBC connection properties. You must also ensure that the solidDB JDBC Driver has access to the solidDB linked library access (LLA) and GSKit libraries. If the JDBC client cannot load the GSKit and LLA libraries, the login data for an externally authenticated user cannot be verified.

About this task

You can install the GSKit and LLA libraries on the client computer in two ways:

- Use the solidDB installation program to install the solidDB package on the client computer. The GSKit and LLA libraries are installed during solidDB installation.
- Copy the libraries to the client computer manually. For details, see the instructions below.

Procedure

1. If your client computer does not have a solidDB installation, copy the GSKit library and auxiliary libraries from the computer where you have installed solidDB server to the client computer. The GSKit library names and default installation locations are shown in the table below:

Table 15. GSKit libraries and default installation locations

Platform	GSKit library	GSKit default installation location	Auxiliary library directories
Windows	gsk8iccs.dll gsk8iccs_64.dll	<solidDB installation directory>\bin	<solidDB installation directory>\bin\C <solidDB installation directory>\bin\N
Linux	libgsk8iccs.so libgsk8iccs_64.so	<solidDB installation directory>/bin	<solidDB installation directory>/bin/C <solidDB installation directory>/bin/N
Solaris	libgsk8iccs_64.so	<solidDB installation directory>/bin	<solidDB installation directory>/bin/C <solidDB installation directory>/bin/N
HP-UX	libgsk8iccs_64.so	<solidDB installation directory>/bin	<solidDB installation directory>/bin/C <solidDB installation directory>/bin/N
AIX	libgsk8iccs_64.so	<solidDB installation directory>/bin	<solidDB installation directory>/bin/C <solidDB installation directory>/bin/N

2. Copy the LLA library from the computer where you have installed solidDB server to the client computer. The LLA library names and default installation locations are shown in the table below:

Table 16. Linked library access (LLA) system libraries

Platform	Dynamic LLA library
Windows	`bin\ssolidacxx.dll`
AIX	`lib/libssolidacxx.so` This is a symbolic link that gives you access to the actual library file `bin/ssolidacxx.so`.
HP-UX	`lib/libssolidacxx.so` This is a symbolic link that gives you access to the actual library file `bin/ssolidacxx.so`.
Linux	`lib/libssolidacxx.so` This is a symbolic link that gives you access to the actual library file `bin/ssolidacxx.so`
Solaris	`lib/libssolidacxx.so` This is a symbolic link that gives you access to the actual library file `bin/ssolidacxx.so`

xx is the version number of the driver library, for example, `ssolidac70.so`.

3. Add the location of the LLA library to the LD_LIBRARY_PATH or LIBPATH (Linux and UNIX) or PATH (Windows) environment variable.

   Note:

   - In Linux and UNIX environments, you need to link to the symbolic link library `libssolidacxx` in the `/lib` directory. Alternatively, rename the `ssolidacxx` library in the `/bin` directory as `libssolidacxx`.
   - In Windows environments, the LLA library is located in the `\bin` directory.
   - In Linux and UNIX environments, use the following syntax:

     ```
     export LD_LIBRARY_PATH=<path_to_library>:$LD_LIBRARY_PATH
     ```

     For example:

     ```
     export LD_LIBRARY_PATH=home/admin/IBM/soliddb/soliddb7.0/lib:$LD_LIBRARY_PATH
     ```

     or

     in AIX environments:

     ```
     export LIBPATH=<path to library>:$LIBPATH
     ```

     For example:

     ```
     export LIBPATH=home/admin/IBM/soliddb/soliddb7.0/lib:$LIBPATH
     ```

   - In Windows environments, use the following syntax:

     ```
     set PATH=<path_to_library>;%PATH%
     ```

     For example:

     ```
     set PATH=C:\soliddb\bin;%PATH%
     set PATH="C:\Program Files\IBM\solidDB\solidDB7.0\bin";%PATH%
     ```

4. Set the connection property `solid_use_strong_encryption` to `yes`.
5. Set the connection property `solid_gskit_path` to point to the directory where the GSKit library is installed. Use the operating system conventions for defining the directory path.

Example

Example: External authentication settings in Windows environments when connecting with Driver Manager

```
set PATH=C:\solid_client\bin;%PATH%

Properties props = new Properties();
// enable GSKit encryption
props.put("solid_use_strong_encryption", "yes");
// define GSKit library path
props.put("solid_gskit_path", "C:\\solid_client\\bin");
```

Example: External authentication settings in AIX environments when defining connection properties in the connect string

The following example enables the use of GSKit by defining connection property in the connect string. The GSKit library path is defined in the PATH environment variable.

```
export LIBPATH=home/admin/solid_client/lib:$LIBPATH

Connection c = DriverManager.getConnection
("jdbc:solid://9.11.22.314:1315//admin?T3stus3r?
solid_use_strong_encryption=yes?solid_gskit_path=home/admin/solid_client/bin");
```

Related tasks:

"Example: Configuring external authentication for JDBC connections - Windows" on page 73
This example showcases the configuration steps need for authenticating solidDB users using the Windows operating system provided authentication mechanism. The external authentication functionality is configured and tested by modifying a JDBC sample shipped with solidDB.

"Installing and configuring IBM Global Security Kit (GSKit) for external authentication – server" on page 59
To use external authentication, you must install the IBM Global Security Kit (GSKit) on the solidDB server computer. If the solidDB server cannot access the GSKit library, the login data for an externally authenticated user cannot be verified. On the server side, the use of GSKit for encryption is controlled with the **General.UseGSKit** parameter.

Configuring your system for external authentication

To use external authentication on Linux and UNIX systems, you need to configure your system so that solidDB can authenticate users using the Pluggable Authentication Modules (PAM) mechanism. On Windows systems, you should define the default domain name for the externally authenticated users. You also need to enable the use of IBM Global Security Kit (GSKit) on both the server and client computers.

Configuring external authentication on AIX systems:
About this task

The following procedure describes a typical way of configuring your AIX system for using external authentication with the solidDB server. The procedure assumes that you have installed and created the necessary Pluggable Authentication Modules (PAM) on your system. For more information about using PAM on AIX, see the AIX 7.1 Information Center.

Important: Misconfigured PAM settings can cause an abnormal shutdown of the solidDB server. To address any problems with authentication, test the external authentication settings in a testing environment first.

Procedure

1. Define the solidDB service name with the **General.PamServiceName** parameter.

   The **General.PamServiceName** parameter defines the solidDB program name that is used in the PAM configuration to define how solidDB users are authenticated. The factory value of **General.PamServiceName** is `solid`.

2. Edit the PAM configuration file at `/etc/pam.conf`. Add the following lines to the file:

   ```
   <service_name>    auth        required     <module_path>
   <service_name>    account     required     <module_path>
   <service_name>    password    required     <module_path>
   <service_name>    session     required     <module_path>
   ```

   where

   - *service_name* defines the name of the solidDB service, as defined by the **General.PamServiceName** parameter.
   - *module_path* defines the name and path of the authentication module.

Examples

If the **General.PamServiceName** parameter value is `solid` (default) and the authentication module you have installed and created on your system is `/usr/lib/security/pam_ldap`, add the following lines in the `/etc/pam.conf` file:

```
solid           auth          required     pam_ldap
solid           account       required     pam_ldap
solid           password      required     pam_ldap
solid           session       required     pam_ldap
```

Instead of using custom-made PAM modules, you can use the `pam_aix` authentication module that is typically included in AIX installations. For example:

```
solid           auth          required     pam_aix use_new_state
solid           account       required     pam_aix
solid           password      required     pam_aix
solid           session       required     pam_aix
```

However, when using the `pam_aix` module, the following limitations apply:

- You must run solidDB as an administrator (root user). To connect to solidDB from a client running on an AIX system, the user does not need to have administrator rights.
- The service name used in the `/etc/pam.conf` file must match the value of the **General.PamServiceName** parameter. If the entries do not match, the system uses the default PAM settings, which can cause an abnormal shutdown of the solidDB server.

Configuring external authentication on HP-UX systems:
About this task

The following procedure describes a typical way of configuring your HP-UX system for using external authentication with the solidDB server. The procedure assumes that you have installed and created the necessary Pluggable Authentication Modules (PAM) modules on your system.

Important: Misconfigured PAM settings can cause an abnormal shutdown of the solidDB server. To address any problems with authentication, test the external authentication settings in a testing environment first.

Procedure

1. Define the solidDB service name with the **General.PamServiceName** parameter.

   The **General.PamServiceName** parameter defines the solidDB program name that is used in the PAM configuration to define how solidDB users are authenticated. The factory value of **General.PamServiceName** is `solid`.

2. Edit the PAM configuration file at `/etc/pam.conf`. Add the following lines to the file:

```
<service_name>      auth  required        libpam_hpsec.so.1
<service_name>      auth  required        libpam_ldap.so.1
<service_name>      account  required      libpam_hpsec.so.1
<service_name>      account  required      libpam_ldap.so.1
<service_name>      password  required     libpam_hpsec.so.1
<service_name>      password  required     libpam_ldap.so.1
<service_name>      session  required      libpam_hpsec.so.1
<service_name>      session  sufficient    libpam_ldap.so.1
```

   where

   - *service_name* defines the name of the solidDB service, as defined by the **General.PamServiceName** parameter.

Examples

If the **General.PamServiceName** parameter value is `solid` (default), add the following lines in the `/etc/pam.conf` file:

```
solid      auth  required        libpam_hpsec.so.1
solid      auth  required        libpam_ldap.so.1
solid      account  required      libpam_hpsec.so.1
solid      account  required      libpam_ldap.so.1
solid      password  required     libpam_hpsec.so.1
solid      password  required     libpam_ldap.so.1
solid      session  required      libpam_hpsec.so.1
solid      session  sufficient    libpam_ldap.so.1
```

Configuring external authentication on Linux systems:
About this task

The following procedure describes a typical way of configuring your Linux system for using external authentication with the solidDB server. The procedure assumes that you have installed and created the necessary Pluggable Authentication Modules (PAM) modules on your system.

Important: Misconfigured PAM settings can cause an abnormal shutdown of the solidDB server. To address any problems with authentication, test the external authentication settings in a testing environment first.

Procedure

1. Define the solidDB service name with the **General.PamServiceName** parameter.

   The **General.PamServiceName** parameter defines the solidDB program name that is used in the PAM configuration to define how solidDB users are authenticated. The factory value of **General.PamServiceName** is `solid`.

2. Create a file in the `/etc/pam.d/` directory. Name the file with the same service name as defined with the **General.PamServiceName** parameter. Add the following lines to the file:

```
#%PAM-1.0
auth  include  system-auth
```

Examples

If the **General.PamServiceName** parameter value is `solid` (default), create a file named `solid` in the `/etc/pam.d` directory.

Configuring external authentication on Solaris systems:
About this task

The following procedure describes a typical way of configuring your Solaris system for using external authentication with the solidDB server. The procedure assumes that you have installed and created the necessary Pluggable Authentication Modules (PAM) modules on your system. The configuration instructions assume your system is set up to use LDAP authentication through PAM.

Important: Misconfigured PAM settings can cause an abnormal shutdown of the solidDB server. To address any problems with authentication, test the external authentication settings in a testing environment first.

Procedure

1. Define the solidDB service name with the **General.PamServiceName** parameter.

   The **General.PamServiceName** parameter defines the solidDB program name that is used in the PAM configuration to define how solidDB users are authenticated. The factory value of **General.PamServiceName** is `solid`.
2. Edit the PAM configuration file at `/etc/pam.conf`. Add the following lines to the file:

   ```
   <service_name>   auth   requisite  pam_authtok_get.so.1
   <service_name>   auth   required  pam_dhkeys.so.1
   <service_name>   auth   required  pam_unix_cred.so.1
   <service_name>   auth   sufficient  pam_unix_auth.so.1
   <service_name>   auth   required  pam_ldap.so.1
   <service_name>   account  required  pam_ldap.so.1
   ```

   where

   - *service_name* defines the name of the solidDB service, as defined by the **General.PamServiceName** parameter.

Examples

If the **General.PamServiceName** parameter value is `solid` (default), add the following lines in the `/etc/pam.conf` file:

```
solid   auth   requisite  pam_authtok_get.so.1
solid   auth   required  pam_dhkeys.so.1
solid   auth   required  pam_unix_cred.so.1
solid   auth   sufficient  pam_unix_auth.so.1
solid   auth   required  pam_ldap.so.1
solid   account  required  pam_ldap.so.1
```

Configuring external authentication on Windows systems:
About this task

The following procedure describes typical configuration steps on a Windows system when using external authentication with the solidDB server. The procedure assumes that your system includes the necessary Security Support Provider Interface (SSPI) services.

On Windows systems, the operating-system-based authentication typically uses a two-part user ID that is composed of a domain and user name such as: `chicago_prod\solid_admin`. In this example, `chicago_prod` is a domain and `solid_admin` is the user name. To ease the use of a two-part user ID, you can use the **General.DefaultDomainName** parameter to specify the domain name that all solidDB users use by default.

When a valid domain name is defined with the **General.DefaultDomainName** parameter, you need to provide only the user name of the externally authenticated users when creating the login credentials. Similarly, externally authenticated users can then log on without specifying the domain name.

The solidDB server uses the value of the **General.DefaultDomainName** parameter to resolve the two-part user ID at connection time.

Defining the default domain is useful for the following reasons:

- When the domain name is defined with the **General.DefaultDomainName** parameter, solidDB stores only the user name of the externally authenticated user in the SYS_USERS table. For example, schema names in your database then default to the one-part user name stored in the SYS_USERS table.
- You can change between the external and internal authentication methods. The domain name for the user accounts that were created to use internal authentication can be specified with the **General.DefaultDomainName** parameter without the need to modify the user name.

Note:

Alternatively, you can leave the **General.DefaultDomainName** parameter empty (default) and provide the domain name as part of the user ID of each externally authenticated user.

Defining default domain name on Windows systems:
Procedure

Define the default domain name with the **General.DefaultDomainName** parameter. The default domain name is the domain name of the computer where your solidDB server is installed.
The **General.DefaultDomainName** parameter does not have a factory value.

Results

When the user enters the user name to authenticate to the system, solidDB uses the value of **General.DefaultDomainName** to resolve the user name as expected by the operating system.

Examples

If the domain name of the server where your solidDB server is running is `chicago_prod`, specify the following setting in the `solid.ini` file:

```
[General]
DefaultDomainName=chicago_prod
```

You can then create the user `solid1` with the CREATE USER statement as follows:

```
CREATE USER solid1 IDENTIFIED EXTERNALLY
```

Defining Windows domain name as part of the user ID:

If you do not specify the domain name with the **General.DefaultDomainName** parameter, you need to provide the Windows domain name as part of the user ID of each externally authenticated user.

Procedure

To define the domain name as part of the user ID, use one of the following formats:
```
domain_name\username
username@domain_name
```

Note:

When using the `CREATE USER user_name EXTERNALLY` statement, the *user_name* string with `\` or `@` character must be given in double quotation marks.

Examples

If the domain name of the server where your solidDB server is running is `chicago_prod` and the user name is `solid1`, create the user using one of the following statements:

```
CREATE USER "chicago_prod\solid1" IDENTIFIED EXTERNALLY
CREATE USER "solid1@chicago_prod" IDENTIFIED EXTERNALLY
```

Creating externally authenticated accounts for database administrators

The external authentication method for a database administrator account must be specified when creating a database. To create a new database with external authentication, use the solidDB startup option **-p** and omit the password.

Before you begin

- The database administrator must have a corresponding operating system or domain level account on the machine where solidDB is running.
- Install and enable GSKit on the server and client computers.
- Configure the external authentication mechanism according to your operating system.
  - On Linux and UNIX systems, you must have the appropriate Pluggable Authentication Module (PAM) service configured in the operating system. See "Configuring your system for external authentication" on page 65 for details.
  - On Windows systems, you must have the appropriate Security Support Provider Interface (SSPI) service configured in the operating system. Also, define the default domain name of the server where solidDB is running with the **General.DefaultDomainName** parameter. See "Configuring external authentication on Windows systems" on page 68.

Procedure

Create a new solidDB database using the following syntax:

```
solid -p -U username -C catalog_name
```

where
username must match the user name of a user that has an operating system user account.

Tip:
If you do not specify a user name or a catalog name, solidDB prompts for them.

Examples

```
solid -p -U soliduser1 -C DBA
```

What to do next

To access solidDB as an externally authenticated user:

1. If you are accessing solidDB from a client computer, ensure that GSKit is enabled on the client computer.
2. Log on using the operating system or domain user account user name and password.

Note: If the database administrator account uses external authentication, you cannot disable the use of GSKit. If the database administrator account is externally authenticated and **General.UseGSKit** is set to no, solidDB server startup fails with the error `External authentication requires GSKit to be enabled`.

Creating externally authenticated accounts for users

To enable the external authentication method for a user, use the CREATE USER or ALTER USER statements. You need to use the keyword EXTERNALLY and omit the password.

Before you begin

- The user must have a corresponding operating system or domain level account on the machine where solidDB is running.
- You must have administrator privileges to enable external authentication for a user.
- Install and enable GSKit on the server and client computers.
- Configure the external authentication mechanism according to your operating system.
  - On Linux and UNIX systems, you must have the appropriate Pluggable Authentication Module (PAM) service configured in the operating system. See "Configuring your system for external authentication" on page 65 for details.
  - On Windows systems, you must have the appropriate Security Support Provider Interface (SSPI) service configured in the operating system. Also, define the default domain name of the server where solidDB is running with the **General.DefaultDomainName** parameter. See "Configuring external authentication on Windows systems" on page 68.

Procedure

Creating a new user account

- To create a user with external authentication, use the following syntax:

  ```
  CREATE USER <username> IDENTIFIED EXTERNALLY
  ```

  where

username must match the user name of a user that has an operating system user account.

Modifying an existing user account

- To change the user account of an existing user to use external authentication, use the following syntax:

  ```
  ALTER USER <username> IDENTIFIED EXTERNALLY
  ```

  where

  username must match the user name of a user that has an operating system user account.

Examples

```
CREATE USER soliduser1 IDENTIFIED EXTERNALLY
ALTER USER soliduser2 IDENTIFIED EXTERNALLY
```

What to do next

To access solidDB as an externally authenticated user:

1. If you are accessing solidDB from a client computer, ensure that GSKit is enabled on the client computer.
2. Log on using the operating system or domain user account user name and password.

Note:

- If the use of GSKit is disabled on the solidDB server (**General.UseGSKit=no**), connections for externally authenticated users fail with error `Error 08004: Server rejected the connection`.
- If the use of GSKit is disabled on the client computer or the client cannot load the GSKit library, connections for externally authenticated users fail with error `SQLAllocEnv`.

Disabling external authentication

To disable the external authentication method for a user, use the ALTER USER statement, specifying the password solidDB uses to authenticate the user internally.

Procedure

To change the user account of an existing user to not use external authentication, use the following syntax:

```
ALTER USER username IDENTIFIED BY password
```

Example

```
ALTER USER soliduser1 IDENTIFIED BY Hippo123
```

Checking authentication type of users

You can check whether a user is authenticated internally or externally by querying the SYS_USERS system table.

Procedure

Use the following command to check the authentication type of all users:

```
SELECT ID, NAME, AUTHENTICATION FROM SYS_USERS;
```

The column AUTHENTICATION contains information about the authentication type of the user:

- 0 – internal authentication
- 1 – external authentication

Example

```
solsql> SELECT ID, NAME, AUTHENTICATION FROM SYS_USERS;
       ID NAME               AUTHENTICATION
       -- ----               --------------
        1 DBA                         0
        2 OMEGA                       1
        9 PELLE                       1
3 rows fetched.
```

Example: Configuring external authentication for JDBC connections - Windows

This example showcases the configuration steps need for authenticating solidDB users using the Windows operating system provided authentication mechanism. The external authentication functionality is configured and tested by modifying a JDBC sample shipped with solidDB.

Before you begin

- solidDB installed in the default directory: `C:\Program Files\IBM\solidDB\solidDB7.0`. The default installation includes the IBM Global Security Kit (GSKit) and linked library access (LLA) libraries and the JDBC sample.
  - GSKit: `bin\gsk8iccs.dll` or `gsk8iccs_64.dll`
  - LLA: `bin\ssolidac70.dll`
  - JDBC sample: `samples\jdbc`
- You can run the solidDB JDBC sample successfully: to compile the sample, you need a working installation of Java Development Kit (JDK) 1.4.2 or newer.

About this task

The example includes the following steps:

- Configuring external authentication for the solidDB server, solidDB tools (and ODBC clients), and JDBC client
- Creating a database with an internally authenticated DBA
- Creating an externally authenticated user using solidDB SQL Editor (**solsql**)

  In the examples, the Windows domain is `chicago` and the user name of the externally authenticated user is `testuser`.
- Compiling a sample application (`samples\jdbc\sample1.java`)
- Connecting to solidDB server with a JDBC connection as an externally authenticated user

Procedure

1. Modify the external authentication related parameters in `solid.ini`.

   Add the following lines to the `solid.ini` configuration file in the `samples\jdbc\run` directory:

   ```
   [General]
   UseGSKit=yes
   GSKitPath=C:\Program Files\IBM\solidDB\solidDB7.0\bin
   DefaultDomainName=<Windows_domain_name>  ;for example: chicago

   [Client]
   ```

```
UseGSKit=yes
GSKitPath="C:\Program Files\IBM\solidDB\solidDB7.0\bin"
;Note: If the path contains a white space,
;enclose the path in double quotation marks.
```

Tip: In this example, the `solid.ini` file in `samples\jdbc\run` functions as both the server-side and client-side configuration file.

However, the `[Client]` section parameters are not needed for JDBC connections. Instead, the parameter settings are needed if you want to test that you can connect to solidDB server with **solsql** as an externally authenticated user (step 5).

2. Check that the location of the GSKit and LLA libraries is defined in the PATH environment variable.

   To add the default installation directory of the GSKit and LLA libraries to PATH, issue the following command:

   ```
   set PATH="C:\Program Files\IBM\solidDB\solidDB7.0\bin";%PATH%
   ```

3. Start solidDB server and create a new database with an internally authenticated DBA with user name `dba` and password `dba`.

   Use the `samples\jdbc\run` as the working directory.

   ```
   cd C:\Program Files\IBM\solidDB\solidDB7.0\samples\jdbc\

   ..\..\bin\solid -c run -Udba -Pdba -Cdba
   ```

   The solidDB server starts, listening at `tcp 2315`.

4. Connect to the solidDB server using the DBA account.

   ```
   ..\..\bin\solsql -c run "tcp 2315" dba dba

   IBM solidDB SQL Editor (teletype) - Version: 7.0.0.2 Build 2012-04-20
   Copyright Oy International Business Machines Ab 1993, 2012.
   Connected to 'tcp 2315'.
   Execute SQL statements terminated by a semicolon.
   Exit by giving command: exit;
   solsql>
   ```

   If the **solsql** connection fails with the error message `SQLAllocEnv`:

   - Check that the **solsql** working directory contains the `solid.ini` file with the **Client.UseGSKit** and **Client.GSKitPath** parameters.
   - Check that the GSKit path is defined correctly with the **Client.GSKitPath** parameter.

5. Create an externally authenticated user `testuser` using **solsql**.

   For example, if the user name for your Windows user account is `testuser`, issue the following command:

   ```
   solsql>CREATE USER testuser IDENTIFIED EXTERNALLY;
   ```

6. Optional: Check the authentication types of the users by querying the SYS_USERS system table.

   For example:

   ```
   solsql> SELECT ID, NAME, AUTHENTICATION FROM SYS_USERS;
          ID NAME              AUTHENTICATION
          -- ----              --------------
           1 DBA                        0
           4 TESTUSER                   1
   2 rows fetched.
   ```

   Value 1 in the column AUTHENTICATION means that the user is authenticated externally.

7. Optional: Restart **solsql** and log in as the externally authenticated user.

```
solsql> quit;
IBM solidDB SQL Editor exiting.

C:\Program Files\IBM\solidDB\solidDB7.0\samples\jdbc\run>..
\..\..\bin\solsql "tcp 3315" testuser T3stus3r
IBM solidDB SQL Editor (teletype) - Version: 7.0.0.2 Build 2012-04-20
Copyright Oy International Business Machines Ab 1993, 2012.
Connected to 'tcp 3315'.
Execute SQL statements terminated by a semicolon.
Exit by giving command: exit;
solsql>
```

8. Modify `sample1.java` by adding the external authentication related JDBC properties.

   Add the GSKit configuration information as new properties:

```
  props = new Properties();
  props.put("StatementCache","32");   // existing property in sample1.java
  props.put("solid_gskit_path",
             "C:\\Program Files\\IBM\\solidDB\\solidDB7.0\\bin");
  props.put("solid_use_strong_encryption", "yes");
```

9. Compile `sample1.java`.

```
C:\Program Files\IBM\solidDB\solidDB7.0\samples\jdbc>javac sample1.java
```

10. Execute the sample application. The application will prompt you to provide the solidDB JDBC connect string, including the user name and password of the externally authenticated user.

    Issue the following command:

```
java -classpath ..\..\jdbc\SolidDriver2.0.jar;. sample1
```

    For example:

```
C:\Program Files\IBM\solidDB\solidDB7.0\samples\jdb
c>java -classpath ..\..\jdbc\SolidDriver2.0.jar;. sample1

JDBC sample application starts...
Application tries to register the driver.
Driver succesfully registered.
Now sample application needs a connectstring in format:

jdbc:solid://<host>:<port>/<user name>/<password>

Please enter the connect string (default:jdbc:solid://localhost:2315/dba/dba)>
```

    For example, provide the following connect string:

```
jdbc:solid://localhost:2315/testuser/T3stus3r
```

    If the login details are correct, the application will continue as follows:

```
Attempting to connect :jdbc:solid://localhost:2315:testuser/T3stus3r

SolidDriver succesfully connected.
Query executed and result set obtained.
Obtaining metadata information.
Metadata information for columns is as follows:
Column i:1  TABLE_SCHEMA,12,WVARCHAR
Column i:2  TABLE_NAME,12,WVARCHAR
...
...
Row 89 : _SYSTEM        SYS_SYNC_REPLICA_PROPERTIES     BASE TABLE
Row 90 : _SYSTEM        SYS_BACKGROUNDJOB_INFO  BASE TABLE
Result set dumped. Sample application finishes.
```

Tip: In some environments, you might need to provide the GSKit and LLA library path with `-Djava.library.path` when starting the application.

For example:

```
java -Djava.library.path=..\..\bin -classpath ..\..\jdbc\SolidDriver2.0.jar;. sample1
```

Related concepts:

4.6.4, "External authentication with Java fails with `java.lang.UnsatisfiedLinkError: ssolidac70`," on page 89

4.2 Encryption

The solidDB server offers two encryption methods for keeping your data secure: the built-in DES algorithm and the IBM Global Security Kit (GSKit). By default, DES encryption is used and only passwords are encrypted. The use of GSKit must be configured separately. If you want to encrypt the database files and log files using DES or GSKit, you need to create an encrypted database using solidDB command-line options. You can also disable the encryption of passwords.

DES algorithm

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 single-DES algorithm that is not recommended for applications that require strong security.

IBM Global Security Kit (GSKit)

The IBM Global Security Kit (GSKit) is a library that can be used with the solidDB server to enforce strong encryption of passwords and data. The GSKit is shipped and installed with the solidDB server. The GSKit library must be available on both the computer where your application (client) is running, and on the computer where solidDB server is running. As with DES encryption, a startup password must be specified when creating, starting, or decrypting a GSKit encrypted database.

GSKit uses the RSA algorithm for public-key encryption. You can set the RSA key length to 1024 (default), 2048, or 4096 bits.

GSKit is supported with the solidDB ODBC Driver and the solidDB data management tools, except for the solidDB SA API based solidDB Speed Loader **solload**. When using GSKit, use the solidDB ODBC API based **solloado**.

4.2.1 Enabling encryption with IBM Global Security Kit (GSKit)

The IBM Global Security Kit (GSKit) library can be used with solidDB to enforce strong encryption of passwords and data. The use of GSKit for encryption is controlled with the **General.UseGSKit** parameter on the server side. If you want to use external authentication, equivalent connection settings are needed also on the client side.

About this task

The GSKit library is installed during solidDB installation. The solidDB installation program installs the GSKit libraries that the solidDB server and clients must be able to load. The GSKit installation includes also a set of auxiliary libraries.

The GSKit libraries for the most common platforms are shown in the table below:

Table 17. GSKit libraries and default installation locations

Platform	GSKit library	GSKit default installation location	Auxiliary library directories
Windows	gsk8iccs.dll gsk8iccs_64.dll	<solidDB installation directory>\bin	<solidDB installation directory>\bin\C <solidDB installation directory>\bin\N
Linux	libgsk8iccs.so libgsk8iccs_64.so	<solidDB installation directory>/bin	<solidDB installation directory>/bin/C <solidDB installation directory>/bin/N
Solaris	libgsk8iccs_64.so	<solidDB installation directory>/bin	<solidDB installation directory>/bin/C <solidDB installation directory>/bin/N
HP-UX	libgsk8iccs_64.so	<solidDB installation directory>/bin	<solidDB installation directory>/bin/C <solidDB installation directory>/bin/N
AIX	libgsk8iccs_64.so	<solidDB installation directory>/bin	<solidDB installation directory>/bin/C <solidDB installation directory>/bin/N

Procedure

1. Ensure that the GSKit library and auxiliary libraries are available on the computer where solidDB server is running.
2. On the server computer:
   a. Set the **General.UseGSKit** parameter to yes.
   b. Set the **General.GSKitPath** parameter to point to the directory where the GSKit library is located.
   c. Optional: If you want that the passwords of any internally authenticated users are sent over a network connection using strong encryption, set the **General.GSKitLoginRequired** parameter to yes.

   For example:

   ```
   [General]
   UseGSKit=yes
   GSKitPath=/home/IBM/solidDB/soliddb-7.0/bin/
   GSKitLoginRequired=yes
   ```
3. Optional: If you want to use external authentication, install and configure GSKit on the client computer.

See “Installing and configuring IBM Global Security Kit (GSKit) for external authentication” on page 59 for details.

Results

The GSKit is used for encryption of passwords.

What to do next

To encrypt a database using GSKit, follow the instructions in 4.2.2, “Encrypting database and log files.”

4.2.2 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>` or `-S <password>`.

Before you begin

The procedure for encrypting database and log files is the same regardless of whether you want to use DES or GSKit encryption.

- To use DES encryption, ensure that **`General.UseEncryption`** is set to `yes` and **`General.UseGSKit`** is set to `no`.
- To use GSKit encryption, ensure that
  - **`General.UseGSKit`** is set to `yes`.
  - **`General.GSKitPath`** is set to point to the directory where the GSKit library is located.

About this task

- The `-E` option in the solidDB startup command invokes database encryption. The database can be encrypted when creating a new database or when starting an existing database.
- An 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 you specify an empty password, the encryption key is left unprotected.
  - The `-x keypwdfile:<filename>` option provides the encryption password from a file.
  - The `-S <encryption_password>` option provides the encryption password as part of the startup command.

    Note: Providing the password within the startup command 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.
- If you want to create an encrypted database with an externally authenticated database administrator, include the `-p` option and omit the `-P <password>` option that specifies the administrator password.

Creating an encrypted database with an internally authenticated database administrator

Procedure

1. To use DES encryption, ensure that **General.UseEncryption** is set to yes (default) and **General.UseGSKit** is set to no (default).

   ```
   [General]
   UseEncryption=yes
   UseGSKit=no
   ```

2. 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 admin123 -E  -x keypwdfile:pwd.txt
   ```

   Tip: Alternatively, you can use the -S <encryption_password> option to specify the encryption password as part of the startup command. For example:

   ```
   solid -C mycatalog -U admin -P admin123 -E  -S admin456
   ```

Creating an encrypted database with an externally authenticated database administrator

Procedure

1. To use GSKit encryption, ensure that **General.UseGSKit** is set to yes and **General.GSKitPath** point to the directory where the GSKit library is located.

   ```
   [General]
   UseGSKit=yes
   GSKitPath=<valid_directory_path>
   ```

2. To create an encrypted database where the database administrator is authenticated externally, include the -p, -E, and -x keypwdfile:<filename> options and omit the -P <password> option in the solidDB startup command.

   For example:

   ```
   solid -p -C DBA -U soliduser1 -E -x keypwdfile:pwd.txt
   ```

   - The option -p specifies that the database administrator with user name soliduser1 is created as an externally authenticated user.

   Tip: Alternatively, you can use the -S <encryption_password> option to specify the encryption password as part of the startup command. For example:

   ```
   solid -p -C DBA -U soliduser1 -E -S admin456
   ```

Encrypting an existing database

Before you begin

The procedure for encrypting database and log files is the same regardless of whether you want to use DES or GSKit encryption.

- To use DES encryption, ensure that **General.UseEncryption** is set to yes and **General.UseGSKit** is set to no.
- To use GSKit encryption, ensure that
  - **General.UseGSKit** is set to yes.
  - **General.GSKitPath** is set to point to the directory where the GSKit library is located.

Procedure

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

```
solid -U admin -P admin123 -E -x keypwdfile:pwd.txt
```

Tip: Alternatively, you can use the `-S <encryption_password>` option to specify the encryption password as part of the startup command. For example:

```
solid -U admin -P admin123 -E -S admin456
```

4.2.3 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 <encryption_password>
```

4.2.4 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 using the command-line option `-S`.

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

4.2.5 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
```

4.2.6 Disabling encryption of passwords

The encryption of passwords can be disabled with server-side or client-side parameters, or at connection time using ODBC Connect Info settings or non-standard JDBC connection properties.

By default, solidDB always encrypts passwords using the DES algorithm. If you have enabled the use of IBM Global Security Kit (GSKit), **General.UseGSKit=yes**, GSKit is used for encryption of passwords of externally authenticated users. Databases and log files are not encrypted by default.

If you want to create a database without any encryption, disable the encryption of passwords using the parameter settings or connection properties described below.

Disabling the encryption of passwords disables also the encryption of database and log files, if used.

Server-side parameter setting

To disable encryption of passwords, set the server-side parameters **General.UseEncryption** and **General.UseGSKit** to no.

```
[General]
UseEncryption=no
UseGSKit=no
```

The default setting for **General.UseEncryption** is yes.

The default setting for **General.UseGSKit** is no.

Client-side parameter setting

To disable the encryption for a specific ODBC client connection, set the client-side parameter **Client.UseEncryption** and **Client.UseGSKit** to no.

```
[Client]
UseEncryption=no
UseGSKit=no
```

The default setting is Yes.

Alternatively, disable encryption using the connect string option USE_ENCRYPTION=NO or USE_GSKIT=NO.

ODBC connect info option

In ODBC environments, disable encryption by including the option USE_ENCRYPTION=NO or USE_GSKIT=NO in the ODBC connect info string.

The option must be given before the server connect string, for example:

```
USE_ENCRYPTION=NO tcp 1964
USE_GSKIT=NO tcp 1964
```

The defaults are USE_ENCRYPTION=YES and USE_GSKIT=NO.

JDBC connection property

In JDBC environments, disable encryption by setting the non-standard JDBC connection property **solid_use_encryption** to NO or **solid_use_strong_encryption** to NO.

4.2.7 Setting RSA key length for GSKit encryption

GSKit uses the RSA algorithm for public-key encryption. You can set the RSA key length to 1024 (default), 2048, or 4096 bits.

About this task

The RSA key length (size in bits) is controlled with the **General.RSAKeySize** parameter. The access mode of the **General.RSAKeySize** parameter is RO (read-only).

Procedure

To modify the RSA key size:

1. Shutdown the server.
2. Modify the **General.RSAKeySize** parameter in the solid.ini configuration file. Valid values are 1024 (default), 2048, or 4096.
3. Restart the server.

Results

If the value of the **General.RSAKeySize** parameter is invalid, the server start fails with the following error message:

```
11046,System,Fatal Error, GSKit enabled, but RSA key size is invalid. Check the value.
```

4.2.8 Querying database encryption status

You can check whether a database is encrypted using the DATABASE_ENCRYPTION_LEVEL() function.

Procedure

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

- 0 - no encryption
- 1 - encrypted

Example

```
solsql> SELECT DATABASE_ENCRYPTION_LEVEL();
DATABASE_ENCRYPTION
-------------------
         0
1 rows fetched.
```

4.2.9 Making backups of encrypted databases

When you create a backup or netbackup of an encrypted database, the backup database is encrypted with the same encryption key and password.

If you use netbackup and your database is encrypted using IBM Global Security Kit (GSKit), the GSKit must be enabled and available on the netbackup server. Also, you might need to restart the netbackup server after making a netbackup of an database that is encrypted with GSKit.

4.2.10 Encrypting HotStandby servers

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

To encrypt HotStandby servers that are using DES encryption:

1. Ensure that encryption is enabled on both the Primary and Secondary servers.
2. Encrypt the database on the Primary server.
3. Netcopy the encrypted database to the Secondary server.
4. Connect the HotStandby servers.

To encrypt HotStandby servers that are using IBM Global Security Kit (GSKit) encryption:

1. Ensure GSKit is enabled and available on both Primary and Secondary servers.
2. Encrypt the database on the Primary server.
3. Netcopy the encrypted database to the Secondary server. The Secondary server shuts down with the error `Encryption password has not been given for encrypted database`.
4. Restart the Secondary server using the same encryption password as used on the Primary.
5. Connect the HotStandby servers.

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

Related tasks:

4.2.1, “Enabling encryption with IBM Global Security Kit (GSKit),” on page 76
The IBM Global Security Kit (GSKit) library can be used with solidDB to enforce strong encryption of passwords and data. The use of GSKit for encryption is controlled with the **`General.UseGSKit`** parameter on the server side. If you want to use external authentication, equivalent connection settings are needed also on the client side.

4.2.2, “Encrypting database and log files,” on page 78
The encryption of the entire database (database and log files) is enabled using command-line options `-E` and `-x keypwdfile:<filename>` or `-S <password>`.

4.2.11 Encryption and performance

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

- On read type operations in disk-based tables, performance impact is mostly determined by the cache hit rate and is not significant when the cache hit rate is high. Encryption has no impact on read operations in in-memory tables.

- On insert and update operations, the server encrypts and decrypts the log files. The performance penalty can be more significant than with read operations.

4.3 Authorization, privileges, and roles

Users can successfully execute operations only if they have the authority to perform the specified function. To create a table, a user must be authorized to create tables; to alter a table, a user must be authorized to alter the table; and so forth. solidDB offers several methods for managing user authorization.

Privileges and roles

A *privilege* is a permission to perform an action or a task. Authorized users can create objects, have access to objects they own, and can pass on privileges on their own objects to other users by using the GRANT statement. Privileges may be granted to individual users or roles (groups).

You can apply five different kinds of user privileges. A user may be able to view, delete, insert, update or reference information in a table or view. Any combination of these privileges may also be applied. A user who has no privileges to a table is not able to use the table at all.

A *role* is a group of privileges that can be granted to users as one unit. You can create roles and assign users to certain roles. A single user may have more than one role assigned, and a single role may have more than one user assigned.

There both system roles and user-defined roles. User-defined roles are created with the `CREATE ROLE`. All roles are granted to users with the `GRANT ROLE` statement.

System roles

solidDB offers the following system roles. The system role names are reserved user names.

Table 18. System roles

Reserved Names	Description
PUBLIC	This role grants privileges to all users. When user privileges to a certain table are granted to the role *PUBLIC*, all current and future users have the specified user privileges to this table. This role is granted automatically to all users.
SYS_ADMIN_ROLE	This is the default role for the database administrator. This role has administration privileges to all tables, indexes and users, as well as the right to use solidDB Remote Control. This is also the database creator role.
_SYSTEM	This is the schema name of all system tables and views.
SYS_CONSOLE_ROLE	This role has the right to use solidDB Remote Control, but does not have other administration privileges.
SYS_SYNC_ADMIN_ROLE	This is the administrator role for data synchronization functions.
SYS_SYNC_REGISTER_ROLE	This role is only for registering and unregistering a replica database to the master.

4.4 Using solidDB with SELinux

SELinux (Security Enhanced Linux) is a security enhancement feature in Linux that provides administrators additional control over which users and applications can access which system resources. solidDB supports SELinux on Red Hat Enterprise Linux (RHEL) operating systems.

Before you begin

The instructions in this section assume that you are familiar with SELinux for RHEL 6. For information about SELinux on RHEL 6, see the Red Hat Enterprise Linux 6 Security-Enhanced Linux User Guide.

You also need to have the following SELinux policy tools installed on your system:

- `selinux-policy-`*`version`* (for example, `selinux-policy-3.7.19-54.el6.noarch`)
- `policycoreutils-python-`*`version`* (for example, `policycoreutils-python-2.0.83-19.1.el6.x86_64`)

About this task

With default installation, all solidDB processes run in an unconfined domain, that is, unconfined users can run solidDB processes without any further action.

The following procedure uses the **`sepolgen`** utility to create and install SELinux policy modules for solidDB so that also confined system level users (`system_u`) can start solidDB processes.

Tip: You need to run the **`sepolgen`** utility separately for each solidDB process.

Procedure

1. In the `selinux/devel` directory, create the policy modules by issuing the following command:

   ```
   sepolgen <solidDB_installation_directory>/bin/<solidDB_executable>
   ```

   The **`sepolgen`** utility creates the policy modules; the file names use the `<solidDB_executable>.xx` naming pattern, for example, `<solidDB_executable>.te`.
2. Install and apply the security policy permanently by issuing the following command:

   ```
   sh <solidDB_executable>.sh
   ```

Results

The **`sepolgen`** utility creates the source and binary files for the policy module. If you want to enforce a more strict policy, for example, for specific users, you need to modify, recompile, and reinstall the policy modules. For more details, see the Red Hat Enterprise Linux 6 Security-Enhanced Linux User Guide.

Examples

Creating and applying the systems default SELinux policy on the solidDB server (`solid`) executable program.

```
 # cd /usr/share/selinux/devel
# secpolgen <solidDB_installation_directory>/bin/solid
# sh solid.sh
```

Creating and applying the systems default SELinux policy on the SMA server (`solidsma`) executable program.

```
 # cd /usr/share/selinux/devel
# secpolgen <solidDB_installation_directory>/bin/solidsma
# sh solidsma.sh
```

Creating and applying the systems default SELinux policy on the solidDB High Availability Controller (`solidhac`) executable program.

```
 # cd /usr/share/selinux/devel
# secpolgen <solidDB_installation_directory>/bin/solidhac
# sh solidhac.sh
```

4.5 Using solidDB audit trail (`AuditTrailEnabled`)

The solidDB audit trail feature enables tracking of user and schema changes persistently within the solidDB database. The audit trail is controlled with the **`Sql.AuditTrailEnabled`** parameter. When audit trail is enabled, information about the database activities are written into a SYS_AUDIT_TRAIL system table. Users with administrator rights can query the SYS_AUDIT_TRAIL system table with normal SQL syntax.

When audit trail is enabled, the system records the following database activities:

- Changes in user and login information
- Changes in schemas and catalogs
- Status of audit trail (enabled/disabled/deletes)

The status of audit trail is written at each server startup. The status message can be used to check when the audit trail data has been collected, and when the server has been started with the audit trail disabled. If auditing is disabled later on, at the next startup, the system writes a status message to indicate that audit trail is disabled.

User access

Only administrators (SYS_ADMIN_ROLE) can query the SYS_AUDIT_TRAIL system table. Administrators are also allowed to DELETE data from the table; the DELETE statements are audited unless the DELETE affected zero rows.

Audit trail and High Availability

In a High Availability setup, only the primary server can write the audit trail. However, audit trail must be enabled in both servers. This is because each server records database activities according to the configuration settings in its own `solid.ini` file. In a switchover (old primary had **`SQL.AuditTrailEnabled=yes`**), the new primary continues to record the changes only if the **`Sql.AuditTrailEnabled`** parameter for it was set to `yes` at the last startup. The state of the new primary is stored as a status message in the system table (AUDIT TRAIL ENABLED (HSB) or AUDIT TRAIL DISABLED (HSB).

4.5.1 Enabling and disabling audit trail

The audit trail is controlled with the **`Sql.AuditTrailEnabled`** parameter. The access mode of the **`Sql.AuditTrailEnabled`** parameter is RO (read-only).

Procedure

- To enable audit trail:

1. Set the **Sql.AuditTrailEnabled** parameter to yes in the solid.ini configuration file.

   ```
   [SQL]
   AuditTrailEnabled=yes
   ```

2. Restart solidDB.

At the startup, the system writes a status message to the SYS_AUDIT_TRAIL system table to indicate that audit trail is enabled. Changes in database activities are recorded in the SYS_AUDIT_TRAIL system table until audit trail is disabled.

- To disable audit trail:
  1. Set the **Sql.AuditTrailEnabled** parameter to no in the solid.ini configuration file.
  2. Restart solidDB.

  At the startup, the system writes a STATUS message to the SYS_AUDIT_TRAIL system table to indicate that audit trail is disabled. Changes in database activities are not recorded in the SYS_AUDIT_TRAIL system table until audit trail is enabled again.

4.5.2 Querying audit trail data in the SYS_AUDIT_TRAIL system table

Users with administrator rights can query the SYS_AUDIT_TRAIL table using normal SQL syntax.

Procedure

- Example: Viewing the SYS_AUDIT_TRAIL system table

```
SELECT CREATIME, LOGIN_USER, SQLSTR FROM sys_audit_trail

           CREATIME LOGIN_USER         SQLSTR
           -------- ----------         ------
2009-03-05 13:21:31 _SYSTEM            AUDIT TRAIL ENABLED
2009-03-05 13:21:42 DBA                CREATE USER DBUSER IDENTIFIED BY
2009-03-05 13:23:13 DBA                CREATE SCHEMA DBA2
2009-03-05 13:23:23 DBA                DROP SCHEMA DBA2
2009-03-05 13:23:24 DBA                CREATE USER DBA2 IDENTIFIED BY
2009-03-05 13:32:22 DBUSER             CREATE TABLE TEST (ID INTEGER)
2009-03-05 13:49:37 DBA                CREATE CATALOG DBUSER
2009-03-05 13:49:59 DBUSER             CREATE TABLE TEST_TAB (ID INTEGER PRIMARY KEY NOT NULL)
```

- Example: Querying CREATE USER operations

```
SELECT CREATIME, LOGIN_USER, SQLSTR FROM sys_audit_trail WHERE type='CREATE USER'"

           CREATIME LOGIN_USER         SQLSTR
           -------- ----------         ------
2009-03-05 13:21:42 DBA                CREATE USER DBUSER IDENTIFIED BY
2009-03-05 13:23:24 DBA                CREATE USER DBA2 IDENTIFIED BY
```

4.6 Troubleshooting encryption and authentication

External authentication requires the use of IBM Global Security Kit (GSKit). If the use of GSKit is not enabled or solidDB server or client cannot load the GSKit library, the server startup or client connection fails.

4.6.1 solidDB server startup fails with error External authentication requires GSKit to be enabled or GSKit enabled, but failed to load the GSKit library

Symptom

The solidDB server startup fails with the following type of errors in solmsg.out and solerr.out:

```
IBM solidDB process has encountered an internal error is unable to
continue normally.
External authentication requires GSKit to be enabled.
...
Server emergency shutdown.
```

```
IBM solidDB process has encountered an internal error is unable to
continue normally.
GSKit enabled, but failed to load the GSKit library. Check the library path.
...
Server emergency shutdown.
```

Causes

- If there is only one database administrator account and the account is externally authenticated and **General.UseGSKit** is set to no, the solidDB server startup fails with the error `External authentication requires GSKit to be enabled`.

  Note: If the database administrator account is authenticated internally but other users are authenticated externally, solidDB server will start even if **General.UseGSKit** is set to no. However, connections for externally authenticated users fails with error `Error 08004: Server rejected the connection`.
- If **General.UseGSKit** is set to yes and solidDB cannot load the GSKit library, the solidDB server startup fails with the error `GSKit enabled, but failed to load the GSKit library. Check the library path.`

Recovery

1. Check that IBM Global Security Kit (GSKit) is installed on the server computer.
2. Check that server-side `solid.ini` file contains the following parameter settings:

   ```
   [General]
   UseGSKit=yes
   GSKitPath=<valid_path_to_GSKit_library>
   ```

Related tasks:

4.2.1, "Enabling encryption with IBM Global Security Kit (GSKit)," on page 76
The IBM Global Security Kit (GSKit) library can be used with solidDB to enforce strong encryption of passwords and data. The use of GSKit for encryption is controlled with the **General.UseGSKit** parameter on the server side. If you want to use external authentication, equivalent connection settings are needed also on the client side.

4.6.2 Connection for an externally authenticated user fails at SQLAllocEnv

Symptom

The connection from solidDB ODBC Driver or solidDB data management tools (such as solidDB SQL Editor (**solsql**)) for an externally authenticated user fails at the function call SQLAllocEnv.

Causes

To use external authentication, the use of IBM Global Security Kit (GSKit) must be enabled and the solidDB client must be able to load the GSKit library. If the solidDB client cannot access the GSKit library, the login data for an externally authenticated user cannot be verified.

Recovery

1. Check that IBM Global Security Kit (GSKit) is installed on the client computer.

2. Check that the client-side `solid.ini` file contains the following parameter settings:

   ```
   [Client]
   UseGSKit=yes
   GSKitPath=<valid_path_to_GSKit_library>
   ```

   The path to GSKit library must be provided using the conventions of your operating system. For example, in Windows environments, if the path contains white space characters, the path must be enclosed in double quotations marks.

   ```
   GSKitPath="C:\Program Files\IBM\solidDB\solidDB7.0\bin"
   ```

or,

1. Set the **`Client.UseGsKit`** parameter to `no`.
2. Log in to solidDB as an internally authenticated user.

Related tasks:

"Installing and configuring IBM Global Security Kit (GSKit) for external authentication – ODBC clients and solidDB tools" on page 61
If you are using the solidDB ODBC Driver or solidDB data management tools (for example, solidDB SQL Editor (**`solsql`**)), to use external authentication, you must install the IBM Global Security Kit (GSKit) on the solidDB client computer. If the solidDB client cannot access the GSKit library, the login data for an externally authenticated user cannot be verified.

4.6.3 Connection for an externally authenticated user fails with `Error 08004: Server rejected the connection`

Symptom

The client connection for an externally authenticated user fails with `Error 08004: Server rejected the connection`.

Causes

- The user name or password you entered was incorrect.
- To use external authentication, the use of IBM Global Security Kit (GSKit) must be enabled on both the server and client computer. If the use of GSKit is not enabled on the server side, solidDB server rejects the connections from externally authenticated users.

Recovery

1. Check that IBM Global Security Kit (GSKit) is installed on the server computer.
2. Check that server-side `solid.ini` file contains the following parameter settings:

   ```
   [General]
   UseGSKit=yes
   GSKitPath=<valid_path_to_GSKit_library>
   ```

4.6.4 External authentication with Java fails with `java.lang.UnsatisfiedLinkError: ssolidac70`

Symptom

The connection from solidDB JDBC Driver for an externally authenticated user fails with the following type of Java exception.

```
Exception in thread "main" java.lang.UnsatisfiedLinkError:
ssolidac70 (Not found in java.library.path)
        at java.lang.ClassLoader.loadLibraryWithPath(ClassLoader.java:995)
        at java.lang.ClassLoader.loadLibraryWithClassLoader(ClassLoader.java:959)
```

```
at java.lang.System.loadLibrary(System.java:453)
at solid.jdbc.SolidConnection.loadDll(Unknown Source)
at solid.jdbc.SolidConnection.<init>(Unknown Source)
at solid.jdbc.SolidDriver.connect(Unknown Source)
at java.sql.DriverManager.getConnection(DriverManager.java:572)
at java.sql.DriverManager.getConnection(DriverManager.java:165)
at sample1.main(sample1.java:79)
```

Causes

To use external authentication, the use of IBM Global Security Kit (GSKit) must be enabled and the solidDB JDBC client must be able to load the GSKit and the linked library access (LLA) libraries.

Recovery

- Check that the bit level of your Java installation is the same as the bit level of your solidDB installation. For example, a 64-bit LLA library does not work with 32-bit Java.
- Check that IBM Global Security Kit (GSKit) and LLA libraries are installed on the client computer.
- Check that you have defined the location of the LLA library correctly.

  For example, in Windows environments, if the path contains a white space character, the path must be enclosed in double quotation marks.

  ```
  set PATH="C:\Program Files\IBM\solidDB\solidDB7.0\bin";%PATH%
  ```

5 Monitoring solidDB

The solidDB server provides various tools for gathering information about database server activity and database operations.

- **ADMIN COMMAND 'userlist'**

  The **ADMIN COMMAND 'userlist -l'** command displays a list of users currently logged in to the database. The output provides information about various database operations and settings for each user.

- **ADMIN COMMAND 'report'**

  The **ADMIN COMMAND 'report'** command produces a report that contains information about the server, users, and database operations. The report also includes the configuration file (`solid.ini`) settings and a list of the performance counters.

- **ADMIN COMMAND 'pmon'**

  The **ADMIN COMMAND 'pmon'** command displays the solidDB performance counters (called *perfmons* or *pmons*) that provide information about various database operations and performance

- **ADMIN COMMAND 'status'**

  The **ADMIN COMMAND 'status'** command displays statistics information about memory usage, process size, transaction count, cache count, user count, database operations.

- **ADMIN COMMAND 'monitor'**

  The **ADMIN COMMAND 'monitor'** command controls monitoring of user activity and SQL calls. The information is logged into the `soltrace.out` file. Monitoring can also be turned on with the command-line option **-m** at solidDB startup.

- **ADMIN COMMAND 'trace'**

  The **ADMIN COMMAND 'trace'** command controls the solidDB trace facility.

- **ADMIN COMMAND 'sqllist'**

  The **ADMIN COMMAND 'sqllist'** command displays a list of the longest running SQL statements among the currently running statements. You can limit the number of statements shown by specifying the number of statements as an attribute (**ADMIN COMMAND 'sqllist top <no_of_statements>'**).

- **ADMIN COMMAND 'backuplist'**

  The **ADMIN COMMAND 'backuplist'** command displays the status of the last local backup.

- **ADMIN COMMAND 'proctrace'**

  The **ADMIN COMMAND 'proctrace'** command controls tracing in stored procedures and triggers.

- **EXPLAIN PLAN FOR**

  The **EXPLAIN PLAN FOR** SQL statement shows the execution plan that the SQL optimizer has selected for a given SQL statement.

- ODBC Driver Manager trace facility (Windows)

  The Windows ODBC Driver Manager has a trace facility that allows the sequence of function calls made by an ODBC application to be recorded into a log file.

5.1 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'`.

5.1.1 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 unrecoverable (fatal) errors, typically causing the server to shut down abnormally

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 increases performance and reduces disk space usage. However, in most cases the improvement is minimal. Disabling generation of log files 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 limited and the message log file accumulates indefinitely without being deleted.

5.1.2 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 E, "Error codes," on page 277.

5.1.3 Using trace files

You can collect various type of trace information about the database operations. The trace facilities are not enable by default as typically you do not need to monitor the trace files for everyday operation of the server. The trace information is needed primarily for troubleshooting of exceptional events. When enabled, the trace information is output to a soltrace.out file in the solidDB working directory.

Related concepts:

"Tracing SQL statements" on page 184
You can trace SQL statements using the ADMIN COMMAND 'trace' and ADMIN COMMAND 'monitor' commands or by using the SQL Info facility.

Related tasks:

"Using stack trace facility" on page 189
The stack traces facility collects diagnostics information upon server failures. In general, IBM Software Support and development teams use the stack traces facility for troubleshooting. You can also generate stack traces to gain information about a problem that you are investigating, but its use is rather limited without knowledge of the solidDB source code.

Related information:

"Tracing communication between client and server" on page 190

5.1.4 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 'soliduser1' 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.

5.2 Checking solidDB version

Procedure

To check the version and build of solidDB:

- Issue the following command:

  ```
  ADMIN COMMAND 'version'
  ```

  or
- Check the version in the `solmsg.out` file.

  The version information is added to the `solmsg.out` file every time the server is started.

5.3 Checking solidDB ODBC and JDBC client version

Procedure

To check the version and build of a solidDB client:

- Issue the following command:

  ```
  ADMIN COMMAND 'userlist -l'
  ```

  The client version of each connected user is listed in the output.

  For example:

  ```
  solsql> ADMIN COMMAND 'userlist -l';
         RC TEXT
         -- ----
          0 DBA
          0   Id: 4
          0   Tid: 1276
          0   Type: ODBC
          0   Machine: solid1 (127.0.0.1)
          0   Login time: 2012-05-24 15:04:08
          0   Appinfo:
          0   Client version: JDBC 7.0.0.2 Build 2012-05-22
          0   Last activity: 2012-05-24 15:05:19
          0   Autocommit: 0
          0   RPC compression: No compression
      ...
      ...
  ```

- Check the version of the solidDB ODBC Driver library as instructed in "Checking solidDB ODBC Driver version" on page 209.
- Check the version of the solidDB JDBC Driver library as instructed in "Checking solidDB JDBC Driver version" on page 210.

5.4 Checking database status

Use the **ADMIN COMMAND 'status'** command to retrieve generic information about the solidDB server, including statistics information about memory usage, process size, transaction count, cache count, user count, and database operations.

Procedure

To retrieve generic information about the solidDB server, issue the following command:

```
ADMIN COMMAND 'status'
```

For example:

```
solsql> ADMIN COMMAND 'status';
       RC TEXT
       -- ----
        0 IBM solidDB started at 2012-06-05 11:17:31
        0 Current directory is C:\solidDB7.0\eval_kit\standalone
        0 Using configuration file C:\solidDB7.0\eval_kit\standalone\solid.ini
        0 Memory statistics:
        0     133658 kilobytes
        0 Process size statistics:
        0     Resident set size: 58648 kilobytes
        0     Virtual size: 153276 kilobytes
        0 Transaction count statistics:
        0     Commit Abort Rollback    Total Read-only  Trxbuf  Active Validate
        0      11308     0      686    11994     12805    6568       2        0
        0 Cache count statistics:
        0     Hit rate      Find      Read     Write
        0     100.0         421718    3         130
        0 Database statistics:
        0     Index writes        68580  After last merge  18169
        0     Log writes          89752  After last cp     87255
        0     Active searches         1  Average              2
        0     Database size       8064 kilobytes
        0     Log size            1088 kilobytes
        0 User count statistics:
        0     Current Maximum Total
        0           3       3   627
```

The result set fields are described below:

- `Memory statistics` shows the amount of memory solidDB has allocated from the operating system. This number does not include the size of the solidDB executable itself.
- `Transaction count statistics` shows the number of different transaction operations since startup.
- `Cache count statistics` shows the cache hit rate and number of cache operations since startup. The cache hit rate is typically above 95 percent. If it is below 95 percent, consider increasing the cache size.
- `Database statistics` shows a number of selected database operations since startup.

  Note: The `Index writes - After last merge` shows the size of the solidDB multiversioning storage tree, 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. An excessively large Bonsai Tree can cause performance degradation. For details on how to reduce Bonsai Tree size, see 8.8, “Reducing Bonsai Tree size by committing transactions,” on page 177.

- `User count statistics` shows the current number of connected users, the maximum number of concurrent users since startup, and the total number of user connections since startup.

Related information:

F.1, "ADMIN COMMAND," on page 359

5.5 Obtaining list of connected users

You can obtain a list of currently connected users by issuing the **ADMIN COMMAND 'userlist'** command.

Procedure

To obtain a list of currently connected users, issue the following command:

```
ADMIN COMMAND 'userlist'
```

For example:

```
solsql> ADMIN COMMAND 'userlist';
       RC TEXT
       -- ----
        0 User name:     User id: Tid:  Type:  Machine id:          Login time:          Client version:
        0 SOL            7        8264  Solcon testi123 (156.2.0.1) 2012-06-05 14:16:48   Not available
        0 SAL            10       3892  ODBC   testi456 (127.0.0.1) 2012-06-05 11:17:51   7.0.0.2 Build 2012-05-30
3 rows fetched.
```

The output provides the following information:

- *User name* - The user name of the connected user.
- *User Id* - The user session identification number (userid) within the database. The lifetime of the userid is that of the user session. After the user logs out, the number can be reused.
- *Tid* - The identification number as a 4-digit code of the current user thread in the server.
- *Type* - Client type. Possible values are:
  - *Java*, which refers to a client using JDBC
  - *ODBC*, which refers to a client using ODBC, including solidDB SQL Editor (**solsql**)
  - *Solcon*, which refers to solidDB Remote Control (**solcon**)
- *Machine id* - The client computer name (host name) and its IP address, if available
- *Login time* - The client computer login timestamp
- *Client version* - The version of the JDBC or ODBC client, as of V7.0.0.2 Interim Fix 2.

  Note:
  - The client version information is not available for clients prior to V7.0.0.2 Interim Fix 2 or for solidDB Remote Control (**solcon**) connections.
  - For solidDB SQL Editor (**solsql**) connections, the ODBC client version is shown.
- *Appinfo* - The value of the client computer's environmental variable SOLAPPINFO (ODBC), or the value of JDBC connection property solid_appinfo.

Tip: You can retrieve more detailed information about each user connection by using the `-l` option in the command (**ADMIN COMMAND 'userlist -l'**).

5.6 Disconnecting (throwout) connected users

You can disconnect single or all users using the **ADMIN COMMAND 'throwout'** command.

Procedure

- To disconnect a single user, issue the following command:

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

  Tip: You can query the *user_id* of each connected user with the **ADMIN COMMAND 'userlist'** command.
- To disconnect all users, issue the following command:

  ```
  ADMIN COMMAND 'throwout all'
  ```

Results

The **ADMIN COMMAND 'throwout'** command does not break the connection between a HotStandby Primary and HotStandby Secondary server.

5.7 Querying the status of the most recent backup

To obtain a status of the most recently run local backup, enter the following command in solidDB SQL Editor (**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.

5.8 Producing reports

The `ADMIN COMMAND 'report'` command produces a report that contains information about the server, users, and database operations. The report also includes the configuration file (`solid.ini`) settings and a list of the performance counters. IBM Software Support may ask you to produce the report for troubleshooting purposes.

To create a report about the status of solidDB, issue the following command:

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

Tip: To ensure that the solidDB Support Assistant collects the reports you have generated, append the *report_filename* with the prefix `rep`.

In general, IBM Software Support and development teams use the reports for troubleshooting. IBM Support may ask you to produce the report for

troubleshooting purposes. You can also generate the report to gain information about a problem that you are investigating, but its use might be limited without knowledge of the solidDB source code.

5.9 Performance counters (perfmon)

The solidDB performance counters (*perfmons* or *pmons*) provide information about various database operations and performance. The performance counters are controlled with the `ADMIN COMMAND 'perfmon'` command.

There are three commands for viewing and collecting performance information:

- `ADMIN COMMAND 'perfmon'` returns performance information for the past few minutes at approximately one minute intervals.
- `ADMIN COMMAND 'perfmon diff'` collects performance information at given intervals and outputs it into a file in a comma-separated value format.
- `ADMIN COMMAND 'perfmon timers'` collects information about execution times of database operations such as SQL execute and file operations for each user.

5.9.1 `ADMIN COMMAND 'perfmon'`

The `ADMIN COMMAND 'perfmon'` command returns a result set of all solidDB performance counters. For troubleshooting purposes, execute `ADMIN COMMAND 'perfmon'` during problematic situations or immediately after.

Example output:

```
ADMIN COMMAND 'perfmon';
       RC TEXT
       -- ----
        0 Performance statistics:
        0 Time (sec)                                          30     42    44     30     34     32     32     33    Total
        0 File open                                      :   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0      0.0
        0 File read                                      :   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0      0.0
        0 File write                                     :   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0      0.0
        0 File append                                    :   0.0    0.0    0.1    0.0    0.0    0.0    0.1    0.0      0.0
        0 File flush                                     :   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0      0.0
        0 File lock                                      :   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0      0.0
        0 Cache find                                     :   0.0    0.0    0.5    0.2    0.2    6.1    0.9    0.0      0.4
        0 Cache read                                     :   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0      0.0
        0 Cache write                                    :   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0      0.0
        0 Cache prefetch                                 :   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0      0.0
        0 Cache prefetch wait                            :   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0      0.0
        0 Cache preflush                                 :   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0      0.0
        0 Cache LRU write                                :   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0      0.0
        ...
```

Each column represents a snapshot of the average performance information for approximately 1 minute. The first row `Time (sec)` shows the point in time when the snapshot is taken. The `Total` column shows average information since solidDB was started.

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.

Note: The collection of performance data is not persistent over server restarts. To collect performance data, you must issue `ADMIN COMMAND 'perfmon'` after each server restart.

`ADMIN COMMAND 'perfmon'` options

The `ADMIN COMMAND 'perfmon'` command syntax also has options that allow you to specify output options. For example, you can restrict the output by providing a list of prefixes of counter names `ADMIN COMMAND 'perfmon` *`name_prefix_list`*`'`.

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)                           19    Total
        0 DBE insert                     :    0.0      0.0
        0 DBE delete                     :    0.0      0.0
        0 DBE update                     :    0.0      0.0
        0 DBE fetch                      :    0.0     41.2
        0 DBE dd operation               :      0        0
        0 Db size                        :   8064     8064
        0 Db free size                   :   7440     7440
        0 DB actiongate lock time, latest:      0        0
        0 DB actiongate lock time, sum   :      0        0
        0 DB actiongate lock count       :      0        0
12 rows fetched.
```

For more information about the ADMIN COMMAND 'perfmon' options, see F.1, “ADMIN COMMAND,” on page 359.

5.9.2 ADMIN COMMAND 'perfmon diff' - 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.

Starting continuous performance reporting

To start collecting performance counter information, issue the following command:

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

where

- *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 ms.

If the continuous performance reporting is already enabled, the ADMIN COMMAND 'perfmon diff start' command returns the following result set:

```
       RC TEXT
       -- ----
        0 Pmon diff output already active.
1 rows fetched.
```

Starting continuous performance reporting

To stop the collection of performance data, issue the following command.

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

To check the status of continuous performance reports

Example

To start logging performance counters in to the counter_log.csv file with 2 second interval , issue the following command:

```
ADMIN COMMAND 'pmon diff start counter_log.csv 2000'
```

5.9.3 ADMIN COMMAND 'perfmon timers'

The ADMIN COMMAND 'perfmon timers' command produces information about execution times of database operations such as SQL execute and file operations for each user. The information can be useful, for example, if you need to troubleshoot on detailed level why certain SQL statements take a long time to execute.

The full syntax for the ADMIN COMMAND 'perfmon timers' command is as follows:

```
ADMIN COMMAND 'perform timers {start | stop | list | clear}'
```

where

`start` starts the timers and clears the existing counter values.

`stop` stops the timers and keeps the current counter values.

`list` lists the current counter values.

`clear` clears the current counter values.

The timer information is given in seconds. The values are cumulative since last **perfmon timers start** or **perfmon timers clear**.

The output can be viewed in the console window (**perfmon timers list**) or printed into a report file with **ADMIN COMMAND 'report** *report_name*'. In the report file, the timer information is listed under the section PERFORMANCE TIMERS.

The output lists the execution times for each user, identified with the userid.

Tip: You can query the *user_id* of each connected user with the **ADMIN COMMAND 'userlist'** command.

Additionally, the output includes information for generic users such as `Merge`, `Checkpoint`, and `Unknown user`.

- The generic users such as `Merge` or `Checkpoint` refer to operations that use multiple threads. The value shows the combined time for all the multithreaded operations. The total time of such operations can be larger than the elapsed time.
- The `Unknown user` shows the time for system connections that are typically unlisted.

Example usage

Typically the timers are used in the following way:

1. Start timers with **ADMIN COMMAND 'perfmon timers start'**.
2. Wait for a few minutes (or more).
3. Get current timers with **ADMIN COMMAND 'perfmon timers list'** or **ADMIN COMMAND 'report** *report_name*'.
4. If necessary wait more and go back to step 3 to get current values.
5. Stop timers with **ADMIN COMMAND 'perfmon timers stop'**.

Example output

```
PERFORMANCE TIMERS
------------------
Elapsed time: 100 sec
User id 3:
```

```
    Measure points               4887702
    Network read               26.526069 sec
    Network write               3.175219 sec
    Process                   73.751434 sec
    SQL prepare                 0.000000 sec
    SQL execute                21.525820 sec
    SQL fetch                   0.000000 sec
    SQL reset                   1.066978 sec
    Statement end              39.658965 sec
    Transaction end             0.172837 sec
    External sorter             0.000000 sec
    Log write                   0.001236 sec
    Buffer pool access          1.852455 sec
    Logreader get data          0.000000 sec
    Logreader scan data         0.000000 sec
    Logreader catchup           0.000000 sec
    Logreader log flush         0.000000 sec
    Logreader spm wait          0.000000 sec
    HSB spm wait                0.000000 sec
    HSB catchup spm wait        0.000000 sec
    File read                   0.000000 sec
    File write                  0.000000 sec
    File flush                  0.000000 sec
    Pages gate wait             0.000000 sec
    Index gate wait             0.000000 sec
    Action gate wait            0.032684 sec
    Merge gate wait             3.574909 sec
    Storage gate wait           0.436523 sec
    Bonsai gate wait            0.068944 sec
    Pessimistic gate wait       0.000000 sec
    Gate wait                   0.000000 sec
Merge:
    Measure points               157382
    Network read                0.000000 sec
    Network write               0.000000 sec
    Process                   38.819534 sec
    SQL prepare                 0.000000 sec
    SQL execute                 0.000000 sec
    SQL fetch                   0.000000 sec
    SQL reset                   0.000000 sec
    Statement end               0.000000 sec
    Transaction end             0.000000 sec
    External sorter             0.000000 sec
    Log write                   0.000000 sec
    Buffer pool access          0.108429 sec
    Logreader get data          0.000000 sec
    Logreader scan data         0.000000 sec
    Logreader catchup           0.000000 sec
    Logreader log flush         0.000000 sec
    Logreader spm wait          0.000000 sec
    HSB spm wait                0.000000 sec
    HSB catchup spm wait        0.000000 sec
    File read                   0.000000 sec
    File write                  0.000000 sec
    File flush                  0.000000 sec
    Pages gate wait             0.000000 sec
    Index gate wait             0.000000 sec
    Action gate wait            0.000000 sec
    Merge gate wait            18.988183 sec
    Storage gate wait           0.000000 sec
    Bonsai gate wait            0.007830 sec
    Pessimistic gate wait       0.000000 sec
    Gate wait                   0.000000 sec
Checkpoint:
    Measure points                     4
    Network read                0.000000 sec
    Network write               0.000000 sec
```

```
    Process                     0.033755 sec
    SQL prepare                 0.000000 sec
    SQL execute                 0.000000 sec
    SQL fetch                   0.000000 sec
    SQL reset                   0.000000 sec
    Statement end               0.000000 sec
    Transaction end             0.000000 sec
    External sorter             0.000000 sec
    Log write                   0.000000 sec
    Buffer pool access          0.000000 sec
    Logreader get data          0.000000 sec
    Logreader scan data         0.000000 sec
    Logreader catchup           0.000000 sec
    Logreader log flush         0.000000 sec
    Logreader spm wait          0.000000 sec
    HSB spm wait                0.000000 sec
    HSB catchup spm wait        0.000000 sec
    File read                   0.000000 sec
    File write                  0.000000 sec
    File flush                  0.000000 sec
    Pages gate wait             0.000000 sec
    Index gate wait             0.000000 sec
    Action gate wait            0.000325 sec
    Merge gate wait             0.000000 sec
    Storage gate wait           0.000000 sec
    Bonsai gate wait            0.000000 sec
    Pessimistic gate wait       0.000000 sec
    Gate wait                   0.000000 sec
Unknown user:
    Measure points               2212640
    Network read                0.000000 sec
    Network write               0.000000 sec
    Process                     0.000000 sec
    SQL prepare                 0.000000 sec
    SQL execute                 0.000000 sec
    SQL fetch                   0.000000 sec
    SQL reset                   0.000000 sec
    Statement end               0.000000 sec
    Transaction end             0.000000 sec
    External sorter             0.000000 sec
    Log write                   0.017517 sec
    Buffer pool access          1.953398 sec
    Logreader get data          0.000000 sec
    Logreader scan data         0.000000 sec
    Logreader catchup           0.000000 sec
    Logreader log flush         0.000000 sec
    Logreader spm wait          0.000000 sec
    HSB spm wait                0.000000 sec
    HSB catchup spm wait        0.000000 sec
    File read                   0.024386 sec
    File write                  9.445144 sec
    File flush                  0.991168 sec
    Pages gate wait             0.000000 sec
    Index gate wait             0.000000 sec
    Action gate wait            0.000000 sec
    Merge gate wait             0.000000 sec
    Storage gate wait           0.222046 sec
    Bonsai gate wait            0.104902 sec
    Pessimistic gate wait       0.000000 sec
    Gate wait                   0.033613 sec
```

5.9.4 List of perfmon counters

The counters are listed in the order they appear in the `ADMIN COMMAND 'pmon'` output.

Table 19. Perfmon counters

Perfmon variable	Description
Time (sec)	In onetime 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 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.
Cache preflush bytes	Number of bytes written by preflusher before log file is flushed. The counter is reset at each flush.
Cache preflush flush	Number of preflush calls/sec before log file is flushed.
RPC messages	Total number of sent messages/sec
RPC read	Total number of read messages/s

Table 19. Perfmon counters (continued)

Perfmon variable	Description
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
RPC connected	Number of client connect requests
RPC disconnected	Number of client disconnect requests
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
SQL direct execute	SQL execute statements/sec using direct statement execute
SQL execute simple	Number of simple SQL statement executes A simple SQL statement is a statement that accesses a single table and does not contain joins, subqueries, function calls, ORDER BY or GROUP BY constructions, and all WHERE conditions are combined with an AND logical operator.
SQL execute complex	Number of complex SQL executes
SQL direct fetch	SQL fetch statements/sec using direct statement execute
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 fetch M-table	Number of rows fetched from in-memory tables
DBE fetch D-table	Number of rows fetched from disk-based tables
DBE dd operation	Server has executed SQL data dictionary operation.
Parallel fetch	Number of rows fetched using parallel operations
Parallel fetch wait	Number of waits for user connections when using parallel operations
Parallel threads active	Number of threads used for parallel operations
Proc compile	Number of procedure compilations
Proc exec	Procedure executions/sec
Proc SQL prepare	Number of SQL prepare calls from a procedure code
Proc SQL execute	Number of SQL execute calls from a procedure code
Proc SQL fetch	Number of SQL fetch calls from a procedure code
Trig compile	Number of trigger compilations
Trig exec	Trigger executions/sec

Table 19. Perfmon counters (continued)

Perfmon variable	Description
Trig SQL prepare	Number of SQL prepare calls from a trigger code
Trig SQL execute	Number of SQL execute calls from a trigger code
Trig SQL fetch	Number of SQL fetch calls from a trigger code
SA fetch	SA-level row fetches/sec
SA insert	SA-level row inserts/sec
SA delete	SA-level row deletes/sec
SA update	SA-level row updates/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
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)
Search active	Table engine-level active searches.
Search replan	Number of search replans A search is replanned when table content is significantly changed. Replan is done to make sure that search plans are optimal for the changed table content.
Db size	Total database size on disk, in KB
Db free size	Free space in the database (page level), in KB

Table 19. Perfmon counters (continued)

Perfmon variable	Description
Mem size	Total size of dynamically allocated memory, in KB
Mem page alloc	Number of explicitly allocated pages
Mem page free	Number of free pages
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
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	Checkpoint status Value 0 means checkpoint is not active. Values 1 and above means checkpoint is active; values above 1 indicate the progress of the checkpoint.
Checkpoint count	Checkpoint serial number from startup
Checkpoint file write	Checkpoint file writes/sec
Checkpoint file read	Checkpoint file reads/sec
Est read samples	Estimator sample refresh call/s
Sorter start sort	Number of external sorts started
Sorter add row	Number of rows added to external sorter
Sorter fetch row	Number of rows read from external sorter
Sorter open file	Number of files opened/sec in external sorter
Sorter activecnt	Number of currently active external sorts
Sorter waitcnt	Number of external sort requests waiting to be started.
Sorter wait	Number of external sort requests waits/sec.
Sorter filecnt	Number of temporary files currently used for external sorting.

Table 19. Perfmon counters (continued)

Perfmon variable	Description
Sorter memblockcnt	Number of memory blocks currently used for external sorting.
Sorter failed	Number of time external sort has failed to start.
Sync repl msg forw	Replica: forwarded 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 write	Log record writes/sec
Log file write	Log block writes/sec
Log file write bytes	Number of log block writes in bytes before log file is flushed
Log nocp write	Pending log records since last checkpoint
Log size	Total size of log file, in KB
Log flush (L)	Logical log flushes/sec (for example, 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 full byt (byte size)	Log writes while log write queue full (in bytes)
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 availq items	Number of records added to available items queue
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 items allocated count	Number of write queue items in the system
Log writeq bytes allocated count	Amount of memory in bytes allocated for write queue items in the system
Log writeq items freed	Internal use only.
Log writeq items blocking waits	Internal use only.
Log writeq items only distribute	Internal use only.
Log writeq remove abort stmt	Number of aborted statements removed from the log queue

Table 19. Perfmon counters (continued)

Perfmon variable	Description
Log writeq remove abort trx	Number of aborted transactions removed from the log queue
Log grpcommits	Number of transactions in the most recent group commit Transaction commits are grouped in one log burst which is written to the log file in a single write.
Log grpcommits (phase2)	Number of HotStandby phase 2 transactions in the most recent group commit. **Note:** You can use this counter only in HotStandby setups.
HSB operation count	Primary/Secondary: transferred log record/sec
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 last catchup recs	Size of HSB catchup in number of sent log records The counter is reset when the catchup is started. You can use this pmon to monitor the progress of the catchup. After catchup is complete, the value shows the size of the completed catchup.
HSB catchup reqcnt	HSB log write requests/sec, for catchup
HSB catchup waitcnt	HSB log waits-for-write requests/sec, for catchup

Table 19. Perfmon counters (continued)

Perfmon variable	Description
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
HSB grpcommits	Number of transactions in the most recent group commit Transaction commits are grouped in one log burst which is send to Secondary as one packet. You can only use this counter on the Primary.
HSB phase1 wait	Internal use only.
HSB secondary ops in packet	Number of log records the Secondary received from the Primary in the most recent log record packet.
HSB secondary trx count	Number of open transactions the Secondary has received from the Primary
HSB secondary locks	Number of row-level locks on the Secondary
HSB secondary lock reqs	Number of lock requests on the Secondary
HSB secondary lock waits	Number of lock waits on the Secondary since the server was started
HSB secondary op waits	Number of times operations (transactions) on the Secondary have been waiting to continue execution
HSB secondary buffers	Number of buffered log record packets the Secondary has received from the Primary
HSB secondary serial mode count	Number of times the Secondary parallel executor has switched to serial mode instead of running in parallel
HSB secondary dispatch queuelength	Size of the most recent dispatch thread (operations to dispatch) on the Secondary
Tabcur create	Number of internal table cursor calls
Tabcur reset full	Number of full constraint reset calls in table cursor
Tabcur reset smpl	Number of simple constraint reset calls in table cursor
Tabcur estimate	Number of cost estimate calls in table cursor
Tabcur cached estimate	Number of table cursor cost estimates found from cached estimates
Tabcur table scan	Number of table scans executed in SQL statements. A high number of table scans can mean that SQL statements are not executed optimally or some index definitions are missing from tables.
Tabcur index access	Number of index accesses executed in SQL statements A high number of index accesses in comparison to number of table scans usually means that SQL statements are properly optimized and correct indexes are defined for tables.
Thread count	Current number of threads

Table 19. Perfmon counters (continued)

Perfmon variable	Description
Trans wait readlvl	Waits/sec for read level at commit *Trans wait readlvl* is a counter that is incremented every time a transaction needs to wait for global read level to become sufficiently high so that the transaction changes become visible (to others) at commit. In regular load situations, this is instantaneous and no wait is needed. In high load situations, a short wait loop might be required. The value of this counter is never decremented. Minor increments (single digits) during 30 second pmon interval are only an indication of a short high-load situation in the server.
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.
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. MME locks
MME max num of locks	Peak number of MME locks (since startup)
MME cur num of lock chains	Current no. MME hash buckets
MME max num of lock chains	Peak no. MME hash buckets (since startup)
MME longest lock chain path	MME: longest hash overflow path
MME mem used by tuples	MME memory allocated to tuples in kilobytes
MME mem used by indexes	MME memory allocated to indexes in kilobytes
MME mem used by page structs	MME memory allocated to the shadow structures in kilobytes
MME page splits	Number of MME page splits
MME page joins	Number of MME page joins
MME unnec mutexed searches	Number of MME rows fetched while unnecessarily in exclusive mode
MME nonmatched (RO)	Number of MME rows that did not match search criteria fetched in shared mode
MME nonmatched (EXCL)	Number of MME rows that did not match search criteria fetched in exclusive mode
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.
MME deletes with x gate	Number of MME deletes performed in exclusive mode
MME hotspot protection	Number of times an MME search enters exclusive mode to access a hotspot

Table 19. Perfmon counters (continued)

Perfmon variable	Description
MME index key inserts	Number of keys inserted to MME indexes, includes keys inserted during a database recovery (not accurate[1])
MME index key deletes	Number of keys deleted from MME indexes. (not accurate[1])
MME bnode resizes	Number of times a MME bnode has been resized
MME vtrie mutex collisions	Number of times optimistic mutexing in vtrie has collided (not accurate[1], congestion[2])
MME vtrie version colls	Number of times a version check in vtrie has collided (not accurate[1], congestion[2])
MME vtrie vertical splits	Number of times a search path in vtrie has been vertically split by a key insert (not accurate[1])
MME vtrie new branches	Number of times a new branch has been added to a vtrie node (not accurate[1]) The approximate branching factor of the vtrie can be calculated as (MME vtrie new branches - MME vtrie branch deletes) / (MME vtrie vertical splits - MME vtrie vertical joins) + 2. This branching factor is only for the vtrie part of the index, the bnode leaf level branching factor cannot be estimated.
MME vtrie vertical joins	Number of times a key delete from vtrie has caused a node on the search path to be deleted (not accurate[1])
MME vtrie branch deletes	Number of times a key delete from vtrie has caused a branch to be removed from a vtrie node (not accurate[1])
MME vtrie search retries	Number of vtrie search retries per second
MME vtrie insert retries	Number of times a vtrie insert has been retried because of a collision (not accurate[1], congestion[2])
MME vtrie delete retries	Number of times a vtrie delete has been retried because of a collision (not accurate[1], congestion[2])
MME bnode mutex collisions	Number of times bnode accesses have caused a mutex collision (not accurate[1], congestion[2])
MME bnode version colls	Number of times bnode accesses have failed because of a version collision (not accurate[1], congestion[2])
MME purge invalidations	Number of purge invalidations per second
MME ffmem privctx init	Internal use only
MME ffmem privctx done	Internal use only
MME ffmem purge	How frequently in seconds the purge routine is called
MME ffmem purge step	How frequently in seconds the purge step routine is called
Posted events queue	Number of posted events that have 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

Table 19. Perfmon counters (continued)

Perfmon variable	Description
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 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.
B-tree node build mismatch	A new mismatch index 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.
B-tree node split	Number of B-tree node splits/sec
B-tree node join	Number of joined B-tree nodes
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 storage fill factor	
B-tree node Bonsai fill factor	
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.
Storage tree height	Current storage 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.

Table 19. Perfmon counters (continued)

Perfmon variable	Description
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.
Action gate wait	Number of action gait waits
MME pages gate wait	Number of gate waits when accessing pages in MME storage
MME index gate wait	Number of gate waits when accessing MME index
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 min	Minimum value of log reader space manager free space (number of operations that can be buffered). Each log reader cursor has its own free space counter; if there are multiple open log reader cursors, the value is the minimum value of free space of all open cursors. If the free space of any log reader cursor is zero, the value of this counter is zero and transaction throttling (slowdown) is enacted.
Logreader spm freespc max	Maximum value of log reader space manager free space (number of operations that can be buffered). If there are multiple open log reader cursors, the value is the maximum value of free space of all open cursors.
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.
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.

Table 19. Perfmon counters (continued)

Perfmon variable	Description
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 InfoSphere® CDC instance when solidDB is a target datastore.
Passthrough open connections	Number of SQL passthrough connections to back-end
Passthrough open statements	Number of prepared statements to back-end
Passthrough reads	Number of executed read-type statements that return rows (for example, `SELECT` statements)
Passthrough non reads	Number of executed write-type statements that return rows (for example, `INSERT` statements)
Passthrough commits	Number of committed statements
Passthrough rollbacks	Number of rollback statements
Passthrough result cnv	Number of fetched (read) rows for which conversion between back-end and solidDB data types have been performed. For example, conversion is needed if the data type in the backend is CHAR(5) and VARCHAR in solidDB.
Passthrough param cnv	Number of statements for which conversion between statement parameters have been performed
Passthrough failures	Number of statements that could not be prepared in back-end
Passthrough reprepared	Number of statements that have been reprepared because write-type statements other that INSERT, UPDATE, and DELETE have been executed in the back-end. Repreparation is needed in such cases to ensure that the table definitions have not been changed, which in turn would cause errors with the prepared statements.
Passthrough complex by num non indexed constraints	Number of statements that are passed through based on the parameter **`Passthrough.ComplexNumNonindexedConstr`**
Passthrough complex by num ordered rows	Number of statements that are passed through based on the parameter **`Passthrough.ComplexNumOrderedRows`**
Passthrough complex by num tables	Number of statements that are passed through based on the parameter **`Passthrough.ComplexNumTables`**
XA trans start	Number of XA transactions that have been started
XA trans end	Number of XA transactions that have ended
XA trans resume	Number of XA transactions that have been resumed
XA trans prepare	Number of XA transactions that have been prepared
XA trans commit	Number of XA transactions that have been committed
XA trans rollback	Number of XA transactions that have been rolled back
XA trans forget	Number of XA transactions that have been forgotten
XA trans recover	Number of XA transactions that have been recovered
XA trans active	Number of XA transactions that are active at the time of the query
SMA connection count	Number of SMA connections

Table 19. Perfmon counters (continued)

Perfmon variable	Description
SMA shared memory used	Amount of shared memory used
TC wait read level	Number of times read operations on the Secondary have been waiting to be executed, when using Transparent Connectivity (TC)
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
Latency below 1 ms	Number of statements for which the latency is below 1 millisecond. To enable the collection of latency statistics: 1. Start the collection of latency statistics with the command ADMIN COMMAND 'perfmon timers start'. 2. View the latency statistics with the command ADMIN COMMAND 'perfmon latency'.
Latency below 2 ms	Number of statements for which the latency is below 1 millisecond.
Latency below 4 ms	Number of statements for which the latency is below 2 milliseconds.
Latency below 8 ms	Number of statements for which the latency is below 4 milliseconds.
Latency below 16 ms	Number of statements for which the latency is below 16 milliseconds.
Latency below 32 ms	Number of statements for which the latency is below 32 milliseconds.
Latency below 64 ms	Number of statements for which the latency is below 64 milliseconds.
Latency below 128 ms	Number of statements for which the latency is below 128 milliseconds.
Latency below 256 ms	Number of statements for which the latency is below 256 milliseconds.
Latency below 512 ms	Number of statements for which the latency is below 512 milliseconds.
Latency over 512 ms	Number of statements for which the latency is over 512 milliseconds.
Time sec	Printout time of this pmon in seconds
Cache seg remains default contra request	Number of cache pages (per second) that are allocated to default (unassigned) segment against request
Cache seg remains special contra request	Number of cache pages (per second) that are allocated to assigned segments against request
Cache seg set default, victim from other	Number of cache pages (per second) that came from assigned segment
Cache seg set default, victim from self	Number of cache pages (per second) that came from the same segment
Cache seg set special, victim from other	Number of cache pages (per second) that came from other assigned or unassigned segment
Cache seg set special, victim from self	Number of cache pages (per second) that came from the same assigned segment
Cache seg remains default per request	Number of cache pages (per second) that are allocated to default (unassigned) segment as requested
Cache seg remains special per request	Number of cache pages (per second) that are allocated to default assigned segments as requested

Table 19. Perfmon counters (continued)

Perfmon variable	Description
Cache seg set default, original dropped	Number of cache pages in a dropped segment

[1] Counters marked as *not accurate* are not accurate because they are not mutex-protected for performance reasons.

[2] In counters that are marked as *congestion*, large increases imply that there is congestion in parallel access when several threads are updating same parts of the database at the same time.

6 Managing network connections

Applications can connect to the solidDB server using network drivers or by linking to the server directly. The solidDB product supports multiple network protocols and connection types simultaneously.

In network-based access methods, the applications and the solidDB server are separate programs, typically communicating using the solidDB ODBC Driver or solidDB JDBC Driver.

Direct linking is provided through linked library access (LLA) and shared memory access (SMA). SMA and LLA are implemented as library files that contain a complete copy of the solidDB server in a library form.

Both the database server and the client applications can connect concurrently to multiple sites using multiple different network protocols. However, some operating systems might limit the number of concurrent users to a single solidDB server process.

6.1 Communication between client and server

The database server and the client transfer information between each other through the computers network communication protocol. The connection between the server and the client is defined with a *network name*. The server listens to the network using certain protocols and server names or port numbers. Clients must use a matching connect string when connecting to the server.

At the server side, the network name is defined as a *network listening name* that identifies the server in the network. When a database server process is started, it publishes at least one network listening name. The server starts to listen to the network using the given network listening name. The network listening name is defined with the **Com.Listen** configuration parameter.

At the client side, the network name is defined as an ODBC or JDBC *connect string* that the client process uses to specify which server it connects to. To establish a connection from a client to a server, the client has to know the network listening name of the server and in some cases, also the location of the server in the network.

ODBC clients

For ODBC connections, a default connect string can be defined with the client-side **Com.Connect** configuration parameter. The connect string can also be supplied, for example, at connection time or when configuring data sources with an ODBC driver manager. The network name consists of a *communication protocol*, a possible set of *options*, and a *server name*, which can be, depending on the protocol, a name or a port number, for example, `tcpip 1315` or `nmpipe solid1`.

Tip:

- Because the network listening name and the connect string must match, the generic term *network name* is used for referring to either one as it is the string that defines the connection between the server and the client.

- With the ODBC API, the network name can also be called *servername* (following the ServerName argument in the SQLConnect() function).

JDBC clients

For JDBC connections, you need to use a JDBC connection string (JDBC url) that specifies the hostname of the computer where solidDB server is running and the port number at which the solidDB server is listening for connections.

Related concepts:

6.4, "Connect strings for JDBC clients," on page 125
For JDBC connections to the solidDB server, you need to use a JDBC connection string (JDBC url) that specifies the hostname of the computer where solidDB server is running and the port number at which the solidDB server is listening for connections.

Related information:

6.3, "Connect strings for ODBC clients," on page 121
The network name used by a client is a data source *connect string*. A connect string for clients that use the ODBC API consists of a *communication protocol*, a possible set of *options*, an optional *host computer name*, and a *server name*. By this combination, the client specifies the server it establishes the connection to. The connect string can also be mapped to *logical data source name*.

6.2 Network listening names (Com.Listen)

The network name of a server is a *network listening name* that consists of a *communication protocol* and a *server name (port number)*. This combination identifies the server in the network. The network name is defined with the **Com.Listen** parameter in the `solid.ini` file.

The syntax of the **Com.Listen** parameter and the network listening name is the following:

```
[Com]
Listen = network_listening_name, network_listening_name, ...
```

where

```
network_listening_name = protocol_name [options] server_name | none
```

- [*options*] can be any combination of the following:

Table 20. Network listening name options

Option	Description	Protocol
-4	Specifies that solidDB listens to IPv4 protocol only.	TCP/IP
-6	Specifies that solidDB listens to IPv6 protocol only. In Windows environments, this option is mandatory if IPv6 protocol is used.	TCP/IP

Table 20. Network listening name options (continued)

Option	Description	Protocol

-i*ip_address* | *host_name*	solidDB listens only to the specified IP address or host name.
	
	Listening to to specified IP address or host name is useful in multi-homed systems that support many TCP/IP interfaces or have multiple IP addresses.
	
	Example:
	`[Com]`
	`Listen = tcp -i127.0.0.1 1313`
	
	A server with the above setting 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.
	
	DNS entries can be used instead of IP addresses, for example:
	`[com]`
	`Listen = tcp -ilocalhost 1313`
		TCP/IP

-o*filename*	Turns on the Network trace facility and defines the name of the trace output file See "Network trace facility" on page 190 for details.	All
-p*level*	Sets the highest level at which the clients can use the solidDB Ping facility. For example, if the server side is set to **-p3**, clients applications can run the Ping facility at levels 1, 2, and 3, but not at 4 and 5. See "Ping facility" on page 192 for details.	All
-t	Turns on the Network trace facility See "Network trace facility" on page 190 for details.	All

- *server_name* depends on the communication protocol:
  - In TCP/IP protocol, `server_name` is a service port number, such as '2315'.
  - In other protocols, `server_name` is a name, such as `soliddb` or `chicago_office`.

  For details on the syntax in different communication protocols, see 6.6, "Communication protocols," on page 125.
- `none` means that all listening ports are disabled.

  The value `none` cannot be set with ADMIN COMMAND 'par'.

Note:

- A server may use an unlimited number of network names.
- All components of network names are not case-sensitive.
- When a database server process is started, it publishes the network names that it starts to listen to. This information is also written to the `solmsg.out` file.
- Network names must be unique within one host computer. For example, you cannot run two servers that are both listening to the same TCP/IP port in one host. However, it is possible that the same port number is in use in different hosts.

Example: `solid.ini` entry

```
[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`. The `tcpip` and `nmpipe` are

communication protocols, while 1313 and soliddb are server names.

Factory value for a network name

If the **Listen** parameter is not set in the solid.ini file or if the value is empty, solidDB listens to the following network names by default:

Table 21. **Com.Listen** *factory values*

Platform	Com.Listen factory values
Windows	NmPipe SOLID ShMem SOLID TCP/IP 1964
Linux and UNIX	UPipe SOLID TCP/IP 1964

6.2.1 Viewing supported protocols for the server

All protocols are not supported in all environments and operating systems.

To view supported protocols for your server, use the following command:

```
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:

Example: Viewing supported protocols in Windows environments

```
solsql> ADMIN COMMAND 'protocols';
       RC TEXT
       -- ----
        0 NmPipe     np
        0 TCP/IP     tc
        0 ShMem      sm
3 rows fetched.
```

6.2.2 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.
- Use the following ADMIN COMMAND:

  ```
  ADMIN COMMAND 'parameter -r com.listen';
  ```

  A list of all the currently set network names for the server is displayed.

  Example:

  ```
  ADMIN COMMAND 'parameter com.listen';
         RC TEXT
         -- ----
          0 Com Listen tcpip 2315, tcpip 1315, tcpip 1964
  1 rows fetched.
  ```

6.2.3 Adding and modifying a network name for the server

You can add and modify network names for solidDB server in the following ways:

- To add network names for the server, use the following ADMIN COMMAND:

  ```
  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.

Note: The **`ADMIN COMMAND 'par com.listen=value'`** command does not replace existing network listening names; it appends new listening names to the existing list.

- Modify the **`Com.Listen`** setting in the `solid.ini` file.

  Use a comma (,) to separate network names.

  Example:

  ```
  [Com]
  Listen = tcpip 1313, nmpipe soliddb
  ```

  You must restart the solidDB server to activate the changes.
- To enable a network name temporarily, use the option **`-x listen:<connect-string>`** at solidDB startup, enclosing the network name in double quotation marks.

  Example:

  ```
  solid -x listen:"tcp 2313"
  ```

6.2.4 Removing network name from the server

To remove a network name for the server permanently, modify the **`Com.Listen`** setting in the `solid.ini` file.

You must restart the solidDB server to activate the changes.

6.3 Connect strings for ODBC clients

The network name used by a client is a data source *connect string*. A connect string for clients that use the ODBC API consists of a *communication protocol*, a possible set of *options*, an optional *host computer name*, and a *server name*. By this combination, the client specifies the server it establishes the connection to. The connect string can also be mapped to *logical data source name*.

A default connect string can be defined with the client-side **`Com.Connect`** configuration parameter. The connect string can also be supplied, for example, at connection time or when configuring data sources with an ODBC driver manager.

The same format of the connect string applies to the **`Com.Connect`** parameter as well as to the connect string used by solidDB tools or ODBC applications.

The format of a connect string is the following:

`protocol_name` [*`options`*] [*`host_computer_name`*] *`server_name`*

where

- *`options`* can be any combination of the following:

Table 22. Connect string options

Option	Description	Protocol
-4	Specifies that client connects using IPv4 protocol only.	TCP/IP

Table 22. Connect string options (continued)

Option	Description	Protocol
-6	Specifies that client connects using IPv6 protocol only. In Windows environments, this option is mandatory if IPv6 protocol is used.	TCP/IP
-i*source_address*	Specifies an explicit connecting socket source address for cases where the system default source IP address binding does not meet application needs. *source_address* can be an IP address or a host name.	TCP/IP
-z	Enables data compression for the connection **Important:** • Data compression is not available for HotStandby connections (**HotStandby.Connect**) and NetBackup connections (**ADMIN COMMAND 'netbackup'**). • Data compression for **netcopy** connections cannot be enabled with the **-z** option. Instead, use the **HotStandby.NetcopyRpcCompress=yes** parameter setting.	All
-c *milliseconds*	Specifies the login timeout (the default is operating-system-specific). A login request fails after the specified time has elapsed.	TCP/IP
-r *milliseconds*	Specifies the connection (or read) timeout. A network request fails when no response is received during the time specified. The value 0 (default) sets the timeout to infinite (operating system default timeout applies).	TCP/IP
-o*filename*	Turns on the Network trace facility and defines the name of the trace output file See *Network trace facility* in the *IBM solidDB Administrator Guide* for details.	All
-p*level*	Pings the server at the given level (0-5). Clients can always use the solidDB Ping facility at level 1 (0 is no operation/default). 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. See *Ping facility* in the *IBM solidDB Administrator Guide* for details.	All
-t	Turns on the Network trace facility See *Network trace facility* in the *IBM solidDB Administrator Guide* for details.	All

- *host_computer_name* is needed with TCP/IP and Named Pipes protocols, if the client and server are running on different machines.
- *server_name* depends on the communication protocol:
  - In TCP/IP protocol, server_name is a service port number, such as '2315'.
  - In other protocols, server_name is a name, such as 'soliddb' or 'chicago_office'.

  For details on the syntax in different communication protocols, see *Communication protocols* in the *IBM solidDB Administrator Guide*.

Note:

- The *protocol_name* and the server_name must match the ones that the server is using in its network listening name.
- If given at the connection time, the connect string must be enclosed in double quotation marks.
- All components of the connect string are case insensitive.

Examples

```
[Com]
Connect=tcp -z -c1000 1315
```

```
[Com]
Connect=nmpipe host22 SOLID
```

```
solsql "tcp localhost 1315"
solsql "tcp 192.168.255.1 1315"
rc = SQLConnect(hdbc, "upipe SOLID", (SWORD)SQL_NTS, "dba", 3, "dba", 3);
rc = SQLDriverConnect(hdbc,
                      (SQLHWND)NULL,
                      (SQLCHAR*)"DSN=tcp localhost 1964;UID=dba;PWD=dba",
                      38,
                      out_string,
                      255,
                      &out_length,
                      SQL_DRIVER_NOPROMPT);
```

6.3.1 Default connect string (Com.Connect)

When no network name is specified for the connection, the default connect string is used. The default connect string is defined with the **Com.Connect** parameter in the client-side `solid.ini` configuration file.

The value of the **Com.Connect** parameter is read by all solidDB tools (**solsql** and so on) and client libraries when no network name 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 an ODBC driver manager is used.

If the **Com.Connect** parameter is not found in the `solid.ini` configuration file, the client uses the default value `tcp localhost 1964` (Windows) or `upipe SOLID` (Linux and UNIX) instead. The server-side **Com.Listen** and client-side **Com.Connect** factory values are set so that if the parameter settings are not available in the `solid.ini` file, the application (client) connects always to a local solidDB server that is listening with the default network name. Thus, local communication (inside one machine) does not necessarily need a `solid.ini` configuration file for establishing a connection.

Example

The following parameter setting in the `solid.ini` of the application workstation defines that the application (client) connects using TCP/IP protocol to a solidDB server that is running on a host computer named `spiff` and listening with the name (port number in this case) `1313`.

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

6.3.2 Logical data source names

The solidDB tools and client libraries support logical data source names. Logical data source names can be used for giving a database a descriptive name.

The logical data source name can be mapped to a data source as a 'logical name' and 'connect string' (network name) pair in the following ways:

- **Using the `[Data Sources]` section in the client-side `solid.ini` file**

  The syntax of the parameters is the following:

  ```
  [Data Sources]
  logical_name = connect_string; Description
  ```

  where `Description` can be used for comments on the purpose of the logical name

  Example:

To map a logical name **My_application** to a database that you want to connect using TCP/IP, include the following lines in the `solid.ini` file:

```
[Data Sources]
My_application = tcpip irix 1313; Sample data source
```

When an application calls the data source 'My_application', the solidDB client maps this to a call to 'tcpip irix 1313'.

- **In Windows environments, using the registry settings (ODBC Driver Manager)**

  You can use the **Control Panel** > **Administrative Tools** > **Data Sources (ODBC)** dialog or the **Registry Editor** (**regedit**) to add mappings.

  For details, see *Configuring the solidDB ODBC Data Source for Windows* in the *IBM solidDB Programmer Guide*.

  Tip: The solidDB data management tools use the solidDB ODBC API. If you have defined an ODBC Data Source, you can use the logical name source name also when connecting to solidDB server with the solidDB tools.

  For example, if you have created a data source named 'solid_1' with ServerName 'tcp 2525', you can connect to the solidDB server with solidDB SQL Editor (**solsql**) using the following command:

  ```
  solsql solid_1
  ```

When connecting to the solidDB server, if the network name is not a valid connect string, the solidDB tools and clients assume it is a logical data source name. To find a mapping between the logical data source name and a valid connect string, the solidDB tools and clients check the client-side `solid.ini` file.

In Windows environments, if the `solid.ini` file is not found or the logical data source name is not defined in the `[Data Sources]` section, the data source settings made with the Windows registry settings are checked in the following order.

1. Look for the Data Source Name from the following registry path:

   ```
   HKEY_CURRENT_USER\software\odbc\odbc.ini\DSN
   ```

2. Look for the Data Source Name from the following registry path

   ```
   HKEY_LOCAL_MACHINE\software\odbc\odbc.ini\DSN
   ```

The check for the logical data source mappings might impact performance:

- If the file system is particularly slow, for example, because the working directory is mapped to a network drive, checking the existence of the `solid.ini` file can have a measurable performance impact.
- In Windows environments, all logical data source mappings in the ODBC registry are checked. The time consumed for this operation is proportional to the amount of defined data sources.
  - With only few (1 to 5) data sources, the connection time will be approximately 5 ms.
  - With 1000 data sources, the connection time will be approximately 200 ms.

  However, if the `solid.ini` file contains the logical data source name mapping, the tools and clients do not try to access the ODBC registry for the mapping.

6.4 Connect strings for JDBC clients

For JDBC connections to the solidDB server, you need to use a JDBC connection string (JDBC url) that specifies the hostname of the computer where solidDB server is running and the port number at which the solidDB server is listening for connections.

The syntax of the JDBC URL (connection string) for the solidDB server is:

```
jdbc:solid://<hostname>:<port>/<username>/<password>[?<property-name>=<value>]...
```

The `port` must match the port number that the server is using in its network listening name (**`Com.Listen`**).

For example:

```
#server-side solid.ini
[Com]
Listen = tcp 2315
```

```
jdbc:solid://localhost:2315/dba/dba
```

For more information about JDBC connections, see *solidDB JDBC Driver* in IBM solidDB Programmer Guide.

6.5 Direct linking with shared memory access (SMA) and linked library access (LLA)

The solidDB server provides a capability to link your application to the server directly, without the need to communicate through network protocols such as TCP/IP. With shared memory access (SMA), you can link multiple applications to a single server instance. With linked library access (LLA), you can link one application. By replacing the network connection with local function calls, performance can be improved significantly.

For more information about creating and connecting linked applications, see the *IBM solidDB Shared Memory Access and Linked Library Access User Guide*.

6.6 Communication protocols

The client process and the solidDB server 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 sections describe the supported communication protocols and common environments that may be used. They also describe the required forms of network names for the various protocols.

Tip: You can view the available communication protocols in your system with the `ADMIN COMMAND 'protocols'` command.

6.6.1 TCP/IP protocol

solidDB supports both TCP/IPv4 and TCP/IPv6 protocols. To use the TCP/IP protocol, you need to specify `tcp` as the protocol, specify the host computer (optional), and use a non-reserved port number.

There are differences in the use of the TCP/IPv4 and TCP/IPv6 protocols, depending on the platform.

- In Linux and UNIX environments, solidDB can listen to both the TCP/IPv4 and TCP/IPv6 protocols automatically, based on the format of the IP address in the network name. If the network name does not specify an IP address, solidDB tries to start listening on IPv6 (::0) first, if it is not possible, it tries again on IPv4 (0.0.0.0).

  If you want solidDB to listen to only one protocol type, you can specify the protocol explicitly with the `-4` (IPv4) and `-6` (IPv6) option in the network name.
- In Windows environments, solidDB listens to the IPv4 protocol by default.

  To use the IPv6, you need to specify the IPv6 protocol using the option `-6` in the network name.

*Table 23. TCP/IP protocol in the network listening name (**Com.Listen**)*

Platform	IPv4 syntax	IPv6 syntax

Linux and UNIX	`Listen = tcp [-4] [-i`*`host_computer`*`] `*`port_number`*
	Examples:
	`Listen = tcp 1315`
	`Listen = tcp -i9.11.22.314 1315`
		`Listen = tcp [-6] [-i`*`host_computer`*`] `*`port_number`*
		Examples:
		`Listen = tcp 1315`
		`Listen = tcp -ife80::9:1122::0314 1315`

Windows	`Listen = tcp [-4] [-i`*`host_computer`*`] `*`port_number`*
	Examples:
	`Listen = tcp 1315`
	`Listen = tcp -i9.11.22.314 1315`
		`Listen = tcp -6 [-i`*`host_computer`*`] `*`port_number`*
		Examples:
		`Listen = tcp -6 1315`
		`Listen = tcp -6 -ife80::9:1122::0314 1315`

*Table 24. TCP/IP protocol in the client connect string (**Com.Connect**)*

Platform	IPv4 syntax	IPv6 syntax

Linux and UNIX	`Connect = tcp [-4] [`*`host_computer`*`] `*`port_number`*
	Examples:
	`Connect = tcp 1315`
	`Connect = tcp 9.11.22.314 1315`
	`Connect = tcpip -4 accounting_dept_server 1315`
		`Connect = tcp [-6] [`*`host_computer`*`] `*`port_number`*
		Examples:
		`Connect = tcp 1315`
		`Connect = tcp fe80::9:1122::0314 1315`
		`Connect = tcpip accounting_dept_server 1315`

Windows	`Connect = tcp [-4] [`*`host_computer`*`] `*`port_number`*
	Examples:
	`Connect = tcp 1315`
	`Connect = tcp 9.11.22.314 1315`
	`Connect = tcpip accounting_dept_server 1315`
		`Connect = tcp -6 [`*`host_computer`*`] `*`port_number`*
		Examples:
		`Connect = tcp -6 1315`
		`Connect = tcp -6 fe80::9:1122::0314 1315`
		`Connect = tcpip -6 accounting_dept_server 1315`

where

`host_computer = ip_address|host_name`

- If the server is running in the same computer with the client program, *host_computer* does not need to be specified.
- If *`host_computer`* is specified as a *`host_name`*, the *`host_name`* must be listed in the `/etc/hosts` file or it must be recognized by the DNS (Domain Name Server).
- If a client attempts to open a TCP/IP connection without specifying a hostname, it uses the local loopback interface address, `127.0.0.1` (IPv4) or `::1` (IPv6) as the default IP address.

`port_number` must be an unreserved port; reserved port numbers are listed in the `/etc/services` file of your system. Select a free number greater than 1024 – smaller numbers are typically reserved for the operating system.

-i *ip_address* or **-i** *host_name* means that the solidDB listens only to the specified IP address or host name. This is useful in multihomed systems that support many TCP/IP interfaces or have multiple IP addresses.

6.6.2 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 good throughput. They are also more secure than TCP/IP, since UNIX Pipes can be accessed only from applications that run on the computer where the server executes.

When using the UNIX Pipes protocol, you must reserve a unique listening name (server name) within the node 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`.

For example, the server 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 server name in uppercase format.

To use the UNIX Pipes protocol, select `upipe` or `unp` as the protocol and enter a server name.

Table 25. UNIX Pipes protocol in the network name

Where	**Syntax example**
Server	`Listen = upipe` *server_name*
Client	`Connect = upipe` *server_name*

Note:

- To use the UNIX Pipes protocol, the server and client processes must run in the same machine.
- The server process must have "write" permission to the directory `/tmp`.
- The client that is accessing UNIX Pipes must have "execute" permission on the directory `/tmp`.
- The directory `/tmp` must exist.

6.6.3 Named Pipes

Named Pipes is a protocol commonly used in the Windows operating systems. To use the Named Pipes protocol, select `nmpipe` or `nmp` as the protocol and enter a server name.

Table 26. Named Pipes protocol in the network name

Where	**Syntax example**
Server	`Listen = nmpipe` *server_name*
Client	`Connect = nmpipe [`*host_computer_name*`]` *server_name*

Note:

- *server_name* must be a character string at most 50 characters long.
- If the server is running in the same computer with the client program, the *host_computer_name* must not be specified.
- If *host_computer_name* is used, the *host_computer_name* must be listed in the`/etc/hosts` file or it must be recognized by the DNS (Domain Name Server).
- To connect to the solidDB server with the Named Pipes protocol, the user must have at least the same rights as the user who started the server.

  For example, if an administrator starts the server, only users with administrator rights are able to connect to the server through Named Pipes. Similarly, if a user with normal user rights starts the server, all users with equal or greater rights are able to connect to the server through Named Pipes.

  If a user does not have proper rights, the solidDB Communication Error 21306 message is given.
- Do not use the Named Pipes protocol with solidDB Remote Control (**`solcon`**); the asynchronous nature of communication between **`solcon`** and the solidDB server may cause problems with the Named Pipes protocol (solidDB server can output messages to **`solcon`** command prompt even though **`solcon`** does not query for such messages explicitly).

6.6.4 Shared Memory

In some cases, the Shared Memory protocol can be the fastest way two processes can exchange information. The Shared Memory protocol 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` as the protocol and enter the server name.

Table 27. Shared Memory protocol in the network name

Where	**Syntax example**
Server	`Listen = shmem server_name`
Client	`Connect = shmem server_name`

Note:

- *server_name* must be a character string less than 128 characters long.
- *server_name* has to be unique only in this computer.

6.6.5 Summary of protocols

The following tables summarize the possible operating systems and required forms for network names for the various communication protocols.

Table 28. solidDB protocols and network names

Protocol	**Server OS**	**Network name in `solid.ini` file**
Named Pipes	Windows	Listen = nmpipe *server*
TCP/IP	Linux, UNIX, Windows	Listen = tcpip *port*

Table 28. solidDB protocols and network names (continued)

Protocol	**Server OS**	**Network name in `solid.ini` file**
UNIX Pipes	Linux, UNIX	Listen = upipe *server*
Shared Memory	Windows	Listen = shmem *server*

Table 29. Application protocols and network names

Protocol	**Server OS**	**Network name in `solid.ini` file**
Named Pipes	Windows	Connect = nmpipe [host] *server*
TCP/IP	Linux, UNIX, Windows	Connect = tcpip [host] *port*
UNIX Pipes	Linux, UNIX	Connect = upipe *server*
Shared Memory	Windows	Connect = shmem *server*

7 Using solidDB data management tools

The solidDB product includes a set of data management tools which are command-line utilities for performing various database tasks.

Console tools

solidDB SQL Editor (`solsql`)
: solidDB SQL Editor (**`solsql`**) is a console tool that you can use to issue SQL statements and solidDB ADMIN COMMANDs at the command prompt. You can also execute script files that contain the SQL statements.

solidDB Remote Control (`solcon`)
: solidDB Remote Control (**`solcon`**) is a console tool for administration; users with administrator rights can issue ADMIN COMMANDs at the command prompt or by executing a script file that contains the commands. With **`solcon`**, the ADMIN COMMANDs can be issued as part of the **`solcon`** startup command line.

  Because only users with administrator rights can access **`solcon`**, if only **`solcon`** is deployed at a production site, the administrators cannot accidentally execute SQL statements that could change the data.

Tools for exporting and loading data

solidDB Speed Loader (`solloado` or `solload`)
: solidDB Speed Loader (**`solloado`** or **`solload`**) loads data from an external file into a database.

solidDB Export (`solexp`)
: solidDB Export (**`solexp`**) exports data from a database into a file. It also creates control files used by solidDB Speed Loader (**`solloado`** or **`solload`**) to perform data load operations.

solidDB Data Dictionary (`soldd`)
: solidDB Data Dictionary (**`soldd`**) exports the data dictionary of a database. It produces an SQL script that contains data definition statements that describe the structure of the database.

Note: solidDB data management tools do not support the Transparent Failover (TF) feature used in High Availability configurations. Transparent Failover hides the server change from the user. For more information, refer to *IBM solidDB High Availability User Guide*.

7.1 solidDB Remote Control (`solcon`)

The solidDB Remote Control (**`solcon`**) is a console tool for administration; it can be used to issue ADMIN COMMANDs at the command prompt or by executing a script file that contains the commands. The ADMIN COMMANDs can be issued also as part of the **`solcon`** startup command line.

Only users with administrator rights can issue commands with **`solcon`**. Because **`solcon`** can be used to issue only ADMIN COMMANDs, it can be useful to deploy only **`solcon`** on a production node; with **`solcon`**, administrators cannot accidentally access or change data in the database by issuing SQL statements.

Important: The ADMIN COMMAND '*command_name*' syntax is different in solidDB Remote Control (**solcon**) and in solidDB SQL Editor (**solsql**). In **solcon**, you must give the command using the *command_name* only, without the prefix ADMIN COMMAND, the single quotation marks, and the line-ending semicolon.

Example: solcon

For example, in **solcon**, you issue the ADMIN COMMAND for backup as `backup`.

```
IBM solidDB Remote Control to "tcp 1964"                    Type help for help

IBM solidDB  - Version 7.0.0.4 Build 2012-12-01
(Windows 64bit MT)
Copyright Oy International Business Machines Ab 1993, 2012.
IBM solidDB 7.0
Exit by giving command: exit

--------------------------------------------------------------------------------
>backup
```

Example: solsql

In **solsql**, you issue the ADMIN COMMAND for backup as `ADMIN COMMAND "backup";`.

solsql:

```
IBM solidDB SQL Editor (teletype) - Version: 7.0.0.4 Build 2012-12-01
Copyright Oy International Business Machines Ab 1993, 2012.
Connected to 'tcp 2315'.
Execute SQL statements terminated by a semicolon.
Exit by giving command: exit;
solsql>ADMIN COMMAND 'backup';
```

7.1.1 Starting solidDB Remote Control (solcon)

Start solidDB Remote Control (**solcon**) with the command `solcon`, followed by argument options.

The syntax for starting **solcon** is:

`solcon [`*options*`][`*network_name*`][`*username*`] [`*password*`]`

where

- *options* can be:

Table 30. **solcon** *command options*

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

- *network_name* is the network name of a solidDB server that you are connected to.

  The given network name must be enclosed in double quotation marks.

Note: Logical data source names can also be used with tools; refer to 6, "Managing network connections," on page 117 for further information.

- *username* is required to identify the user and to determine the user is authorized. Without appropriate rights, command execution is denied.
- *password* is the password of the user for accessing the database.

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

Access rights

To use **solcon**, you must have SYS_ADMIN_ROLE or SYS_CONSOLE_ROLE rights, or the connection is refused.

Error messages

When there is an error in the command line, **solcon** gives you a list of the possible syntax options as a result. Check the command line you entered.

Exiting solcon

To exit **solcon**, enter the command exit.

Examples: solidDB Remote Control (solcon)

Start **solcon** with the server name tcp localhost 1313, the administrator username admin and password iohi4y:

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

Start **solcon** to back up a specific database:

```
solcon -ebackup 'tcpip 1313" dbadmin iohi4y
```

7.1.2 Entering commands in solidDB Remote Control (solcon)

With **solcon**, you can execute ADMIN COMMANDs at the command line with the -e option or in a text file with the -f option.

When you execute administrative commands in solidDB Remote Control, you provide only the ADMIN COMMAND option (*command_name*) as the syntax for the command string, without the quotation marks.

For example, the SQL command **ADMIN COMMAND 'backup'** in solidDB Remote Control is backup.

Table 31. **solcon** *specific commands*

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

Error messages

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

7.2 solidDB SQL Editor (solsql)

The solidDB SQL Editor (**solsql**) is a console tool that is used for issuing SQL statements and ADMIN COMMANDs. The commands and statements can be issued at the command prompt or by executing a script file that contains the SQL statements.

Tip: To access a short description of available ADMIN COMMANDs and their abbreviations, execute the following command:

```
ADMIN COMMAND 'help';
```

7.2.1 Starting solidDB SQL Editor (solsql)

Start solidDB SQL Editor (**solsql**) with the command `solsql`, followed by argument options.

The syntax for starting **solsql** is:

```
solsql [options] [network_name] username [password]
```

where

- `options` can be:

Table 32. **solsql** *command options*

Option Syntax	Description
-a	Autocommit every statement
-c *dir*	Change working directory
-e *sql-string*	Execute the specified SQL string If you use this option, you can only use the **-a** option (autocommit) issue to commit work.
-f *filename*	Execute SQL strings from the specified file Use this option to execute SQL statements programmatically from a script.
-o *filename*	Write result set to the specified file
-O *filename*	Append result set to the specified file
-S *schema_name*	Use only the specified schema
-C *catalog_name*	Use only the specified catalog
-t	Print execution time per command
-tt	Print the time of prepare, execute, and fetch operations per command
-2	Creates two connections to the database You can switch between the two connections with the command `switch`.

Table 32. **solsql** *command options (continued)*

Option Syntax	Description
-u	• In Unicode databases (**General.InternalCharEncoding=UTF8**), expect the data in character and wide character data type columns to be encoded in UTF-8. • In partial Unicode databases (**General.InternalCharEncoding=Raw**), expect the data in wide character data type columns to be encoded in UTF-8. Data in character data type columns is not converted. See section 7.7, "Using solidDB tools with Unicode," on page 158 for more information.
-m	• In Unicode databases (**General.InternalCharEncoding=UTF8**), expect the data in character and wide character data type columns to be encoded in the console's locale/codepage, despite the settings in the server-side and client-side character data binding parameters. • In partial Unicode databases (**General.InternalCharEncoding=Raw**), expect the data in wide character data type columns to be encoded in the console's locale/codepage, despite the settings in the server-side and client-side character data binding parameters. Data in character data type columns is not converted. See section 7.7, "Using solidDB tools with Unicode," on page 158 for more information.
-M *locale_name*	• In Unicode databases (**General.InternalCharEncoding=UTF8**), expect the data in character and wide character data type columns to be encoded in the specified locale/codepage. • In partial Unicode databases (**General.InternalCharEncoding=Raw**), expect the data in wide character data type columns to be encoded in the specified locale/codepage. Data in character data type columns is not converted. The format of *locale_name* depends on the operating system. For example, in Linux environments, the locale name for the code page GB18030 in Chinese/China is zh_CN.gb18030. In Windows environments, the locale name for Latin1 code page in Finnish/Finland is fin_fin.1252. See section 7.7, "Using solidDB tools with Unicode," on page 158 for more information.
-h, -?	Help = Usage
-x onlyresults	Print only the rows of the result set The SQL statement for which the rows are printed must be given using the **-e** *sql-string* option. `solsql -x onlyresults -e "SELECT * FROM customerid" "tcp 2315" dba dba`
-x pwdfile: *filename*	Read password from the file specified with *filename*
-x stoponerror	This option forces a shutdown of **solsql** immediately when an error is detected.
-x returnerroronexit	This option displays return codes for SQL errors and user raised procedure errors. The possible return codes are: • `Code 60: the execution of an SQL statement fails` • `Code 61: a procedure call returns an error` If several SQL statements or procedure calls fail during the execution of an SQL script, the returned code is that of the first failure.
-x outputsql	This option prints out the executed SQL commands instead of only printing out the results of each operation.

- *network_name* is the network name of a solidDB server that you are connected to.

  The given network name must be enclosed in quotation marks. Refer to 6, "Managing network connections," on page 117 for further information.

  Tip: Logical data source names can also be used with the solidDB tools.
- *username* is required to identify the user and to determine the authorization level of the user authorization. Without appropriate rights, command execution is denied.
- *password* is the password of the user for accessing the database. The *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

Note:

- If the *username* and *password* are specified at the command line, the *network_name* must also be specified.
- If the name of the SQL script file is specified at the command line (except with the `-f` option), the *network_name*, *username*, and *password* must also be specified.

Remember to commit work at the end of the SQL script or before exiting **solsql**.

The solidDB tools connect to the first server specified in the **Com.Connect** parameter in the `solid.ini` file. If you specify no arguments, you are prompted for the database administrator user name and password.

Error messages

When there is an error in the command line, **solsql** gives you a list of the possible syntax options as a result. Check that the command line you entered is valid.

Exiting solsql

To exit **solsql**, enter the command `exit`.

Related reference:

7.2.4, "solidDB SQL Editor (**solsql**) commands," on page 137
In addition to SQL statements and ADMIN COMMANDs, there are a number of **solsql** specific commands that you can use to operate **solsql**.

7.2.2 Executing SQL statements with solidDB SQL Editor (solsql)

To issue SQL statements with **solsql**, the statements must be terminated by a semicolon (`;`).

Examples

```
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;
```

7.2.3 Executing an SQL script from a file

You can execute SQL scripts from a file directly in the solidDB SQL Editor or by specifying the script filename in the solidDB SQL Editor startup command line.

Executing an SQL script with `solsql`

The syntax for script calls in **`solsql`** 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 must be relative to the **`solsql`** working directory.

Executing an SQL script from a file at the `solsql` startup

To execute an SQL script from a file at **`solsql`** startup, the name of the script file must be given as a command-line parameter:

```
solsql network_name username password filename
```

All statements in the script must be terminated by a semicolon. **`solsql`** 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 **`solsql`**. If an SQL string is executed with the option `-e`, commit can only be done using the `-a` option.

7.2.4 solidDB SQL Editor (`solsql`) commands

In addition to SQL statements and ADMIN COMMANDs, there are a number of **`solsql`** specific commands that you can use to operate **`solsql`**.

Note: The **`solsql`** commands must be terminated by a semicolon.

Table 33. solidDB SQL Editor (`solsql`) commands

Command	Description
`bye`	Shuts down **`solsql`**
`exit`	Shuts down **`solsql`**
`help`	Displays usage information for **`solsql`**
`quit`	Shuts down **`solsql`**
`solsql_silent` *seconds*	Makes **`solsql`** sleep for *seconds*
`switch`	Switches between the two connections to the database that have been created using the `-2` startup option

7.3 solidDB Speed Loader (solloado and solload)

The solidDB Speed Loader is a tool for loading data from external files into a solidDB database.

There are two variants of the solidDB Speed Loader:

- **solloado** provides support for Unicode and partial Unicode databases. It also enables loading of data with multiple threads. **solloado** is based on the solidDB ODBC API; the client-side configuration parameters can be used to control the behavior of **solloado**.
- **solload** provides support for partial Unicode databases only. **solload** is based on the solidDB SA API.

The solidDB Speed Loader can load data in various 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 data, is specified in a *control file*.

Key characteristics

- The data is loaded into the database through the solidDB engine. This enables online operation of the database during the loading.
- The data to be loaded does not have to be located on the computer as the solidDB server.
- The data is loaded in batches. You can control the number of records committed in one batch.
- The tables must exist in the database in order to perform data loading.
- Catalogs are supported with the following syntax:

  catalog_name.schema_name.table_name
- The following constraints are checked:
  - referential
  - NOT NULL
  - unique
- solidDB Speed Loader does not support check constraints that are defined using the CREATE TABLE and ALTER TABLE statement and specify data value restrictions in columns.

  However, solidDB Speed Loader always checks for unique or foreign key constraints that are defined using the CREATE TABLE statement.

Note: The IBM Global Security Kit (GSKit) is not supported with **solload** connections. When using GSKit, use **solloado**.

7.3.1 File types

The solidDB Speed Loader can load data in various formats. The file that contains the data is called an *import* file. The format of the import file is specified in a control file. Detailed information about the loading process is output into a log file.

Control file

The control file provides information about the structure of the import file. It specifies 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. The solidDB Speed Loader loads data into one table at a time.

Related information:

7.3.5, “Control file syntax,” on page 144

Import file (data file)

The import file is the file that contains the data to be loaded into the solidDB database. The solidDB Export (**solexp**) produces these type of data files.

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 **solexp** 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.

- 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, **solloado** or **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 specified
- Any possible error messages. For details on solidDB Speed Loader errors, see E.31, “solidDB Speed Loader (solloado and solload) errors,” on page 357.

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.

solidDB Speed Loader and `solid.ini` configuration files

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

Client copies of the 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 reads the server name it connects to from the server configuration file.

For example, to connect to a server using the UNIX Pipes protocol and with the server name `solid1`, the following lines are needed in the configuration file:

```
[Com]
Connect=upipe solid1
```

7.3.2 Starting solidDB Speed Loader (`solloado` and `solload`)

Start solidDB Speed Loader with the command `solloado` or `solload`, followed by argument options.

If you start solidDB Speed Loader with no arguments, you see a summary of the arguments with a brief description of their usage.

- The syntax for starting **`solloado`** is:

  `solloado [options] [network_name] username [password] control_file`
- The syntax for starting **`solload`** is:

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

where options can be:

Table 34. **`solloado`** *and* **`solload`** *command options*

Option Syntax	`solloado`	`solload`	Description
`-b` *statements*	X	X	Number of insert statements to commit in one batch (number of statements after which commit is executed) For example, if you specify **`-b 10`**, commit is executed after 10 inserts.
`-B` *records*	X		Number of records to be inserted in 1 statement For example, if you specify **`-B 3`**, each insert inserts 3 rows.
`-c` *dir*	X	X	Change working directory
`-C` *catalog_name*	X	X	Set the default catalog from where data is read from or written to
`-l` *filename*	X	X	Write log entries to this file
`-L` *filename*	X	X	Append log entries to this file
`-m`	X		• In Unicode databases (**`General.InternalCharEncoding=UTF8`**), expect the data in character and wide character data type columns to be encoded in the console's locale/codepage, despite the settings in the server-side and client-side character data binding parameters. • In partial Unicode databases (**`General.InternalCharEncoding=Raw`**), expect the data in wide character data type columns to be encoded in the console's locale/codepage, despite the settings in the server-side and client-side character data binding parameters. Data in character data type columns is not converted. See section 7.7, "Using solidDB tools with Unicode," on page 158 for more information.

Table 34. **solloado** *and* **solload** *command options (continued)*

Option Syntax	solloado	solload	Description
-M *locale_name*	X		• In Unicode databases (**General.InternalCharEncoding=UTF8**), expect the data in character and wide character data type columns to be encoded in the specified locale/codepage. • In partial Unicode databases (**General.InternalCharEncoding=Raw**), expect the data in wide character data type columns to be encoded in the specified locale/codepage. Data in character data type columns is not converted. The format of *locale_name* depends on the operating system. For example, in Linux environments, the locale name for the code page GB18030 in Chinese/China is zh_CN.gb18030. In Windows environments, the locale name for Latin1 code page in Finnish/Finland is fin_fin.1252. See section 7.7, "Using solidDB tools with Unicode," on page 158 for more information.
-u	X		• In Unicode databases (**General.InternalCharEncoding=UTF8**), expect the data in character and wide character data type columns to be encoded in UTF-8. • In partial Unicode databases (**General.InternalCharEncoding=Raw**), expect the data in wide character data type columns to be encoded in UTF-8. Data in character data type columns is not converted. See section 7.7, "Using solidDB tools with Unicode," on page 158 for more information.
-n *records*	X	X	Insert array size (network version)
-s *schema_name*	X	X	Set the default schema
-t	X	X	Print load time
-w *threads*	X		Sets the number of threads inserting data. The value cannot exceed the number of processors Default is 4.
-h	X	X	Help = Usage
-x emptytable	X	X	Load data only if there are no rows in the table
-x errors: *count*	X	X	Maximum error count
-x nointegrity	X	X	No integrity checks during load
-x pwdfile: *filename*	X	X	Read password from the file
-x skip: *records*	X	X	Number of records to skip
-x *utf8*		X	WCHAR data is in UTF-8 format

- *network_name* is the network name of a solidDB server that you are connected to.

  The given network name must be enclosed in quotation marks. Refer to 6, "Managing network connections," on page 117 for further information.

  Tip: Logical data source names can also be used with the solidDB tools.

- *username* is required to identify the user and to determine the authorization level of the user authorization. Without appropriate rights, command execution is denied.
- *password* is the password of the user for accessing the database. The *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

For details on the *control_file*, see section 7.3.5, "Control file syntax," on page 144.

Examples

The following **solloado** example loads data from a file specified by a control file named DBA_TBL.ctr. It reads data as UTF-8 characters, using 8 threads to insert data with 30 records in one statement.

```
solloado -w 8 -B 30  -u "tcpip 1964" dba dba DBA_TBL.ctr
```

The following **solload** example loads data from a file specified by a control file named delim.ctr:

```
solload "tcpip 1964" dba dba delim.ctr
```

Error messages

When there is an error in the command line, **solload** gives you a list of the possible syntax options as a result. Check your command-line entry.

7.3.3 Tips for speeding up loading

To ensure that loading takes place with maximum performance, consider the following aspects:

- Connect locally if possible; it is slower 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.

  To disable logging, set the **Logging.LogEnabled** parameter to no.

  Tip: After the loading has been completed, remember to enable logging again (**Logging.LogEnabled=yes**). Running the server in production use with logging disabled is discouraged. If logs are not written, no recovery can be made if an error occurs due to, for example, power failure or disk error.

7.3.4 Examples of solidDB Speed Loader usage

Example: Loading fixed-format records

In fixed-length format import files, data records have a fixed length and the data fields inside the records have a fixed position and length.

Example: Control File 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)
)
```

Example: Control File 2

```
OPTIONS (SKIP = 10, ERRORS = 5)
-- Skip the first ten records. Stop if
-- error count 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)
```

Example: Loading variable-length records

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

Example: Control File 3

EXAMPLE 1 uses multiple columns that have separators instead of fixed-length fields.

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

Example: Control File 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
-- error count 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 date string
-- given above
-- The default date string is used for DATE2.
-- If the column for DATE2 is 'NULL' a NULL is
-- inserted.
```

Running a sample load using solidDB Speed Loader (solload)

The solidDB package contains a sample that demonstrates how to use **solload** to load files. The sample is available in the samples/importexport/solload directory in your solidDB installation directory.

The sample loads data into a table called TEST1. There are two control files:

- delim.ctr uses delimited fields
- fixed.ctr uses fixed length fields

1. Start the solidDB server.
2. Create a sample table by using the load.sql script and solidDB SQL Editor (**solsql**).
3. Start loading data into the database.
   - To use the delimited fields control file, enter the following command:

     ```
     solload "tcpip 1964" dba dba delim.ctr
     ```

   - To use the fixed-length control file, enter the following command:

     ```
     solload "tcpip 1964" dba dba fixed.ctr
     ```

   The user name and password are assumed to be dba.
4. Verify that the load succeeded. The output of a successful load using delim.ctr or fixed.ctr is:

   ```
   IBM solidDB Speed Loader - Version 6.5.0.4 Build 2011-01-20
   Copyright Oy International Business Machines Ab 1993, 2011.
   Load completed successfully, 19 rows loaded.
   ```

7.3.5 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.

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 can optionally precede the LOAD [DATA] statement.

Table 35. Full syntax of the control file

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

Reserved words

The solidDB Speed Loader reserved words must be enclosed in double quotation marks 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 36. solidDB 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

CHARACTERSET keyword in solidDB Speed Loader

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 as follows:

- ANSI for the ANSI character set
- MSWINDOWS for the Windows character set
- PCOEM for the ordinary PC character set
- IBMPC for the IBM PC character set
- SCAND7BIT for the 7-bit character set containing Scandinavian characters

DATE, TIME, and TIMESTAMP keywords in solidDB Speed Loader

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

- When a keyword 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 that are 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 37. Data masks

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

- Mask parts:
  - Year masks: YYYY and YY
  - Month masks: MM, M, and B (B refers to a three-letter abbreviation (case insensitive) of the month in English)
  - Day masks: DD and D
  - Hour masks: HH and H
  - Minute masks: NN and N
  - Second masks: SS and S
- Masks within a DATE mask may be in any order; for example, the DATE mask could be 'MM-DD-YYYY' (12-18-2010) or 'DD-B-YYYY' (18-DEC-2010).
- 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.
- The masks must be separated

DATE example in Control File

The following example uses the POSITION keyword. For details on this keyword, read "POSITION" on page 151.

```
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

The following example uses the FIELDS TERMINATED BY keyword. For details on this keyword, read "FIELDS TERMINATED BY" on page 150.

```
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 defines delimiting characters around each field. The delimiter can be a single character or two separate characters that precede and follow each data field in the input file. You might use a single character (such as the double quotation mark character) or a pair of characters (such as left and right parentheses) to delimit your fields. If you use double quotation marks as the delimiter and the comma as the terminator/separator, your input might look like the following:

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

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

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

If the keyword OPTIONALLY is used, 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 quotation marks. For example, the following examples have the same effect:

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

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

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

Because using two conventions is potentially confusing, it is not recommended. Instead, use single quotation marks unless you are using a single quotation mark itself as the enclosing character, for example:

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

If you are using single quotation marks as the enclosing characters, you must double the apostrophes as shown in the clause above. For example, to produce `Didn't I warn you?` in the database, the input must be as follows:

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

Almost any printable characters can be used as the "enclosing" characters. The enclosing characters can also be specified using the hexadecimal format. For example, if a hexadecimal string is used, 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, the ENCLOSED BY clause needs to show the character only once. For example, the following clauses have the same effect:

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

The following examples show the input in the control file and the corresponding values 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!""""
```

You can use enclosing characters in the column data itself (embedded field separators). If you use embedded field separators, you can use the TERMINATED BY clause together with the OPTIONALLY ENCLOSED BY clause to ensure that the column data is enclosed correctly. For more information, see "FIELDS TERMINATED BY" on page 150.

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 ")"
```

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 column is of type VARCHAR and the second of type INTEGER.

Treatment of enclosed characters within the data

The ENCLOSED BY characters themselves can occur within the data. However, when occurring within the data, each of the enclosing characters needs to occur twice in the input for each occurrence 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
```

Deeply nested parentheses work the same way. 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 format `American Pie (The Day The Music Died)` in the database, the input file must contain the following:

```
(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 or line feed) can be embedded within a string. For example:

```
(This 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 (colon) is not used in the data and if there is no need to preserve new lines in the input data, only the field separator (not the enclosing characters) is required in the input data.

If your data is fixed-width, 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 quotation marks, for example, ":"
- Surrounded by single quotation marks, for example, ':'
- In hexadecimal format, for example, X'3A'

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

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
```

The trailing colon after the final field is accepted but not required.

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 quotation mark is defined as the character that encloses embedded field separators (commas) in the data file. The OPTIONALLY ENCLOSED BY clause may use either single or double quotation marks to delimit the enclosing characters.

For example:

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

The above illustrates the use of both single and double quotation marks for *enclose_char* in the syntax:

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

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 cannot 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 position of a field 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.

7.4 solidDB Export (solexp)

solidDB Export (**solexp**) is a tool for exporting data from a database into files. solidDB Export produces two types of files for each table:

- **data file** (<tablename>.dat) that contains the exported data
- **control file** (<tablename>.ctr) the specifies the format of the data file

The default file name is the same as the exported table name.

solidDB Speed Loader can use the data and control files to load data into solidDB databases.

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

7.4.1 Starting solidDB Export (solexp)

Start solidDB Export with the command solexp, followed by argument options.

If you start solidDB Export without any arguments, a summary of the arguments with a brief description is displayed.

The syntax for starting **solexp** is:

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

where

- *options* can be:

Table 38. solexp command options

Option Syntax	Description
-c *dir*	Change working directory
-C *catalog_name*	Set the default catalog from where data is read from or written to

Table 38. solexp command options (continued)

Option Syntax	Description
-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
-m	• In Unicode databases (**General.InternalCharEncoding=UTF8**), expect the data in character and wide character data type columns to be encoded in the console's locale/codepage, despite the settings in the server-side and client-side character data binding parameters. • In partial Unicode databases (**General.InternalCharEncoding=Raw**), expect the data in wide character data type columns to be encoded in the console's locale/codepage, despite the settings in the server-side and client-side character data binding parameters. Data in character data type columns is not converted. See section 7.7, "Using solidDB tools with Unicode," on page 158 for more information.
-M *locale_name*	• In Unicode databases (**General.InternalCharEncoding=UTF8**), expect the data in character and wide character data type columns to be encoded in the specified locale/codepage. • In partial Unicode databases (**General.InternalCharEncoding=Raw**), expect the data in wide character data type columns to be encoded in the specified locale/codepage. Data in character data type columns is not converted. The format of *locale_name* depends on the operating system. For example, in Linux environments, the locale name for the code page GB18030 in Chinese/China is zh_CN.gb18030. In Windows environments, the locale name for Latin1 code page in Finnish/Finland is fin_fin.1252. See section 7.7, "Using solidDB tools with Unicode," on page 158 for more information.
-o *filename*	Write exported data to this file This option can be used only when exporting the data of a single table. The default data and control file name is the same as the exported table name (`<tablename>.dat` and `<tablename>.ctr`).
-p	Preserve case of schema and table names
-s *schema_name*	Use only this schema for export
-S	Create SQL insert into clauses
-A	Add attribute names to insert clause

Table 38. solexp command options (continued)

Option Syntax	Description
-u	• In Unicode databases (**General.InternalCharEncoding=UTF8**), expect the data in character and wide character data type columns to be encoded in UTF-8. • In partial Unicode databases (**General.InternalCharEncoding=Raw**), expect the data in wide character data type columns to be encoded in UTF-8. Data in character data type columns is not converted. See section 7.7, "Using solidDB tools with Unicode," on page 158 for more information.
-x pwdfile: *filename*	Read password from the file
-h, -?	Help = Usage

- *network_name* is the network name of a solidDB server that you are connected to.

  The given network name must be enclosed in quotation marks. Refer to 6, "Managing network connections," on page 117 for further information.

  Tip: Logical data source names can also be used with the solidDB tools.
- *username* is required to identify the user and to determine the authorization level of the user authorization. Without appropriate rights, command execution is denied.
- *password* is the password of the user for accessing the database. The *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.

  In some environments you might need to escape the * with double quotation marks ("*").

  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
```

Error messages

- When there is an error in the command line entry, **solexp** gives you a list of the possible syntax 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.

7.5 solidDB Data Dictionary (soldd)

solidDB Data Dictionary (**soldd**) is a tool for retrieving data definition statements from solidDB databases.

soldd produces an 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.

The default file name is `soldd.sql`.

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.

Related concepts:

9.1.7, "Troubleshooting solidDB Data Dictionary (soldd)," on page 202

7.5.1 Starting solidDB Data Dictionary (soldd)

Start solidDB Data Dictionary (**soldd**) with the command `soldd`.

If you start solidDB Data Dictionary with no arguments, you will see a summary of the arguments with a brief description of their usage.

The syntax for starting **soldd** is:

```
soldd [options] [network_name] username [password] {tablename
```

where

- *options* can be:

Table 39. soldd command line options

Option Syntax	Description
-c *dir*	Change working directory
-C *catalog_name*	Set the default catalog from where data definitions are read from or written to
-h, -?	Help = Usage

Table 39. soldd command line options (continued)

Option Syntax	Description
-m	• In Unicode databases (**General.InternalCharEncoding=UTF8**), expect the data in character and wide character data type columns to be encoded in the console's locale/codepage, despite the settings in the server-side and client-side character data binding parameters. • In partial Unicode databases (**General.InternalCharEncoding=Raw**), expect the data in wide character data type columns to be encoded in the console's locale/codepage, despite the settings in the server-side and client-side character data binding parameters. Data in character data type columns is not converted. See section 7.7, "Using solidDB tools with Unicode," on page 158 for more information.
-M *locale_name*	• In Unicode databases (**General.InternalCharEncoding=UTF8**), expect the data in character and wide character data type columns to be encoded in the specified locale/codepage. • In partial Unicode databases (**General.InternalCharEncoding=Raw**), expect the data in wide character data type columns to be encoded in the specified locale/codepage. Data in character data type columns is not converted. The format of *locale_name* depends on the operating system. For example, in Linux environments, the locale name for the code page GB18030 in Chinese/China is zh_CN.gb18030. In Windows environments, the locale name for Latin1 code page in Finnish/Finland is fin_fin.1252. See section 7.7, "Using solidDB tools with Unicode," on page 158 for more information.
-o *filename*	Write data definitions to this file
-O *filename*	Append data definitions to this file
-p	Preserve case of schema and table names
-s *schema_name*	List definitions from this schema only
-u	• In Unicode databases (**General.InternalCharEncoding=UTF8**), expect the data in character and wide character data type columns to be encoded in UTF-8. • In partial Unicode databases (**General.InternalCharEncoding=Raw**), expect the data in wide character data type columns to be encoded in UTF-8. Data in character data type columns is not converted. See section 7.7, "Using solidDB tools with Unicode," on page 158 for more information.
-x tableonly	List table definitions only
-x indexonly	List index definitions only
-x viewonly	List view definitions only
-x sequenceonly	List sequence definitions only

Table 39. soldd command line options (continued)

Option Syntax	Description
-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

- *network_name* is the network name of a solidDB server that you are connected to.

  The given network name must be enclosed in quotation marks. Refer to 6, "Managing network connections," on page 117 for further information.

  Tip: Logical data source names can also be used with the solidDB tools.
- *username* is required to identify the user and to determine the authorization level of the user authorization. Without appropriate rights, command execution is denied.
- *password* is the password of the user for accessing the database. The *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

Note:

- If no table name is given, all definitions to which the user has rights are listed.
- 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.
- The **-t** *tablename* option is still supported in order to keep old scripts valid.

Error messages

When there is an error in the **soldd** startup command line, **soldd** gives you a list of the possible syntax options as a result. 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
```

Related concepts:

9.1.7, "Troubleshooting solidDB Data Dictionary (soldd)," on page 202

7.6 Entering password from a file

User identification information is typically entered as plain text, for example in the solidDB startup command, and in the solidDB data management tools. However, you can enter the password from a file. Entering the password from a file means that it cannot be seen by running the UNIX command **ps**.

The syntax is as follows:

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

where

- *command* can be any of the following:
  - solcon
  - soldd
  - solexp
  - solid
  - solload
  - solloado
  - solsql
- *filename* can be either absolute or relative to the working directory

Password file

In the file where the password is stored, the first character string ending at newline character is read and considered as the password. Preceding space and newline characters are ignored. If the password includes space or newline characters, it must be enclosed in single or double quotation marks. However, using quotation marks means that quotation mark and backslash characters that belong to the password must be escaped by a backslash character.

Examples

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

7.7 Using solidDB tools with Unicode

This section contains information about how to use the solidDB tools with Unicode and non-Unicode databases.

The following solidDB tools can be used to output and import data in the system default locale or a specified locale in both Unicode and partial Unicode databases.

- solidDB SQL Editor (**solsql**)
- solidDB Data Dictionary (**soldd**)
- solidDB Export (**solexp**)
- solidDB Speed Loader (**solloado**)

solidDB Remote Control (**solcon**) does not support conversions of data to UTF-8. For example, if an error message that is output to **solcon** contains Unicode encoded data, it is not displayed correctly in the console.

The locale to be used in conversions is defined with the command line options when starting the tool.

Important:

- The solidDB tools use the solidDB ODBC API 3.5.1; this means that if the binding method for character data types is defined with the server-side **Srv.ODBCDefaultCharBinding** or client-side **Client.ODBCCharBinding** parameters, this setting also impacts the behavior of the solidDB tools.
- The Unicode and partial Unicode databases behave differently in reference to conversions of CHAR and WCHAR data types:
  - **Unicode databases**

    Both CHAR and WCHAR data types are converted between the UTF-8/UTF-16 format in solidDB and the locale/codepage defined with the chosen binding method.
  - **partial Unicode databases**

    CHAR data types are not converted; instead, they are handled in the raw (binary) format that is used to store CHAR data in partial Unicode databases.

    WCHAR data types are converted between the UTF-16 format in solidDB and the locale/codepage defined with the chosen binding method.

Table 40. Command line options for solidDB tools for partial Unicode and Unicode databases

Option	Description
No option/Factory setting	The console locale setting is used, unless overridden with the server-side or client-side parameters in the `solid.ini` file. **Note:** If the server-side **Srv.ODBCDefaultCharBinding** or client-side **Client.ODBCCharBinding** parameter is set to UTF8, the locale of the console must support UTF-8.
`-m`	The console locale setting is used, despite the server-side or client-side parameters in the `solid.ini` file.
`-M<locale_name>`	The locale console setting is overridden with the locale defined with `<locale_name>`. The `<locale_name>` depends on the operating system. For example, in Linux environments, the locale name for the code page GB18030 in Chinese/China is zh_CN.gb18030. In Windows environments, the locale name for Latin1 code page in Finnish/Finland is fin_fin.1252.
`-u`	Input/output is forced to UTF-8.

Note: If the server-side or client-side parameters in the `solid.ini` file are set to use 'Raw' binding, you should always use the `-m`, `-M` or `-u` option to override the `solid.ini` settings.

7.8 Example: Reloading a database using solidDB tools

This example demonstrates how a database can be reloaded to a new one using the solidDB tools.

The database reload procedure can be useful, for example, for minimizing the database file size by removing gaps (unused space) that are created during delete and update operations; the reload rewrites the database without gaps.

Overview:

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 (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 (soldd).**

   Use the following command to extract an SQL script containing definitions for all tables, views, triggers, indexes, procedures, sequences, and events.

   ```
   soldd "tcpip 1964" dbadmin password
   ```

   The **soldd** command lists all data definitions into one SQL file; the default file name is `soldd.sql`.

   Note: User and role definitions are not listed for security reasons. If the database contains users or roles, they must be appended into the extracted SQL file.

2. **All data is extracted with solidDB Export (solexp).**

   Use the following command to extract the control and data files for all tables.

   ```
   solexp "tcpip 1964" dbadmin password *
   ```

   The export creates control files (`table_name.ctr`) and data files (`table_name.dat`) for each table. The default file name is the same as the exported table name. In 16-bit environments, file names longer than eight letters are concatenated.

3. **A new database is created to replace the old one.**

   You can create a replacement database 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 (**solcon**).

   Use the following command to create a backup using solcon:

   ```
   solcon -eBACKUP "tcpip 1964" dbadmin password
   ```

   The option `-e` precedes an administration command.

4. **Data definitions are imported into the new database using the solidDB SQL Editor (solsql).**

   Use the following command to execute the SQL script created by solidDB Data Dictionary (**soldd**).

   ```
   solsql -fSOLDD.SQL "tcpip 1964" dbadmin password
   ```

   This command loads the data definitions 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.

   Tip: The previous two steps can be performed together by starting the solidDB server with the following command:

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

The option `-x` creates a new database, executes commands from a file, and exits. The `-U` and `-P` options define the username and password.

5. **Data is loaded into the new database using the solidDB Speed Loader (`solload`).**

   Use the following command to load data into the new database:

   ```
   solload "tcpip 1964" dbadmin password table_name.ctr
   ```

   Tip: In UNIX environments, the wildcard symbol * can be used.

   To load several tables into the database, a batch file containing a separate command line for each table is recommended.

   The following type of batch files can be used:

   - Shell scripts in UNIX environments
   - `.bat` scripts in Windows environments

8 Performance tuning

This section discusses techniques that you can use to improve the performance of solidDB.

Fast path: 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 213.

- **SQL.IsolationLevel**
- **Logging.DurabilityLevel**

Additionally, using in-memory tables (**General.DefaultStoreIsMemory=yes)** can improve performance significantly.

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

8.1 Logging and transaction durability

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

Standards compliance

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

8.1.1 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 (for example, due to a power failure) 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).

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 are not 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.

With strict durability, the user is not told that the data has been committed until after that data was successfully written to the transaction log on disk. Strict durability 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 **Logging.DurabilityLevel** 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*, 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".

8.1.2 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. Performance over safety is acceptable in situations where, for example, you need to only 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 — for example, 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, your system would still function. Furthermore, 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.

8.1.3 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", 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. If you use relaxed durability, some transactions may not have been written to the log file yet, even though those transactions were committed. Therefore, you should use relaxed durability ONLY when you can afford to lose a small amount of recent data.

If you want to set a maximum delay time before the server writes data, use the **`Logging.RelaxedMaxDelay`** parameter.

8.2 Choosing transaction isolation levels

Concurrency control is based on the application 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. Conflicting transactions are detected 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.

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

8.2.1 Setting the isolation level

By default, the isolation level in the solidDB server is *read committed* (**SQL.IsolationLevel=1**). You can also set the isolation level on transaction level or on session-level using SQL commands.

To set the default isolation level, use the **SQL.IsolationLevel** parameter. The parameter accepts the following values:

- 1 - READ COMMITTED
- 2 - REPEATABLE READ
- 3 - SERIALIZABLE (supported only with disk-based tables)

To set the isolation level for a session, use the following SQL command:

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

For example:

```
SET ISOLATION LEVEL REPEATABLE READ
```

To set the isolation level for a transaction, use the following SQL command:

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

For example:

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

8.3 Controlling memory consumption

The solidDB server allocates main memory 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.

A solidDB server process running in-memory tables is 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*.

Note: Immediately after the server startup, the reported process size in Windows environments is smaller than the actual allocated size. The reported size is smaller because cache pages are allocated at this stage, but they excluded from the process size until they are used for the first time. In Linux and UNIX environments, the cache pages are included. Thus, the reported process size is bigger in Linux and UNIX environments than in Windows environments.

8.3.1 Controlling process size

The process size does not correspond directly to the actual database memory consumption, because the process size contains also non-database elements. The following aspects affect the process size::

- Cache size – The factory value is 32 MB. You can control the database cache size with the **IndexFile.CacheSize** parameter.
- Executable program footprint – The footprint approximately 5 MB, but as different libraries are initializes, the footprint can grow up to 10 MB (varies per platform and release).
- Client threads – Each client consumes a few hundred kilobytes of main memory.
- Dynamic memory reserved for command handling – Server allocates resources for execution plans, temporary data, and so on.
- Statement cache – When the server executes SQL statements, it parses and optimizes them first. This can be time consuming. The server can store the parsed and optimized statements in the virtual memory. The virtual memory allocation is called the statement cache. You can control the statement cache with ODBC and JDBC connection properties.
- The hash table for the transaction lookup table – The **General.LockHashSize** and **MME.LockHashSize** parameters affect the memory consumption. They define the number of elements in the lock hash table.
- Transaction and sort buffers
- Accessed tables that are buffered in the main memory

You can control and monitor the process size by using configuration parameters and ADMIN COMMAND commands. Any violations of process limits you might have set 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.

Srv.ProcessMemoryLimit

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

The factory value for **Srv.ProcessMemoryLimit** is 0; there is no process memory limit. If you use the parameter, set it 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 and transient tables) and the indexes on those in-memory tables
- the amount of memory set aside for the solidDB server's cache (the **IndexFile.CacheSize** 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 **Srv.ProcessMemoryLimit**, the server will accept ADMIN COMMANDs only. You can use the **Srv.ProcessMemoryWarningPercentage** and **Srv.ProcessMemoryLowPercentage** parameters to warn you about increasing process memory consumption.

Note:

- The **Srv.ProcessMemoryLimit** and **Srv.ProcessMemoryCheckInterval** parameters are interlinked; if the **ProcessMemoryCheckInterval** parameter is set to 0, the **ProcessMemoryLimit** parameter is not effective, that is, there is no process memory limit.
- You should not set the **Srv.ProcessMemoryLimit** parameter when using SMA. If you need to limit the memory the SMA server uses, use the **SharedMemoryAccess.MaxSharedMemorySize** parameter.

Srv.ProcessMemoryLowPercentage

The **Srv.ProcessMemoryLowPercentage** parameter sets a warning limit for the total process size. The limit is expressed as percentage of the **Srv.ProcessMemoryLimit** parameter value.

Prior to exceeding the limit, you have exceeded the warning limit defined with the **ProcessMemoryWarningPercentage** parameter and received a warning in the `solmsg.out` log file. When the **Srv.ProcessMemoryLowPercentage** limit is exceeded, a system event is given.

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

Srv.ProcessMemoryWarningPercentage

The **Srv.ProcessMemoryWarningPercentage** parameter sets the first warning limit for the total process size. The warning limit is expressed as percentage of the **Srv.ProcessMemoryLimit** parameter value.

When the **Srv.ProcessMemoryWarningPercentage** limit is exceeded, a system event is given in the `solmsg.out` log file.

The limit set with **Srv.ProcessMemoryWarningPercentage** must be lower than the **Srv.ProcessMemoryLowPercentage** limit.

Srv.ProcessMemoryCheckInterval

The **Srv.ProcessMemoryCheckInterval** parameter defines the interval for checking the process size limits. The interval is given in milliseconds.

The minimum non-zero value for **Srv.ProcessMemoryCheckInterval** is 1000 (ms). Only values 0, 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, the process size checking is disabled.

The **Srv.ProcessMemoryLimit** and **Srv.ProcessMemoryCheckInterval** parameters are interlinked; if the **ProcessMemoryCheckInterval** parameter is set to 0, the **ProcessMemoryLimit** parameter is not effective, that is, there is no process memory limit.

8.3.2 Tuning your operating system

Operating systems store information in:

- real (physical) memory
- virtual memory
- expanded storage
- disk

Your operating system can 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 the system total memory might not be large enough to hold everything for which you have allocated memory.

To improve performance, increase the amount of total memory in your system or decrease the amount of solidDB database cache memory allocated.

Related information:

8.3.3, “Database cache”

8.3.3 Database cache

The information managed by the solidDB server 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. The database cache uses available memory to store information that is read from the hard disk. The database cache is also used to buffer the database pages while the server is executing a checkpoint. When an application next time requests the information that was stored in the cache, the data can be read from memory instead of the hard disk.

Defining database cache size

In a disk-based database, the database cache uses available memory to store information that is read from the hard disk. The database cache is also used to buffer the database pages while the server is executing the checkpoint both in disk-based and in-memory databases. When an application next time requests this information, the data is read from memory instead of from the hard disk.

The default size of the cache depends on the platform. You can change the cache size through the **IndexFile.CacheSize** parameter. Typically, you need to increase the cache size 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 suffices. When you decrease the cache size of an in-memory database, the size cannot be less than 8 MB to facilitate efficient checkpoint activity.

Increase the value of **IndexFile.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 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 whether the server is retrieving most of the data from disk instead of memory by checking the cache hit rate using the command `ADMIN COMMAND 'status'`. Alternatively, check the overall cache and file ratio statistics using `ADMIN COMMAND 'perfmon'`. Typically, the cache hit rate needs to be better than 95%.

Note: If you are using a diskless server with disk-based tables, the database cache size has to be configured to contain the whole database. This is because a diskless server does not use any disk storage space, it maintains all D-tables in the database cache.

Related tasks:

5.4, “Checking database status,” on page 95
Use the **ADMIN COMMAND 'status'** command to retrieve generic information about the solidDB server, including statistics information about memory usage, process size, transaction count, cache count, user count, and database operations.

Related information:

5.9, “Performance counters (perfmon),” on page 98
The solidDB performance counters (*perfmons* or *pmons*) provide information about various database operations and performance. The performance counters are controlled with the `ADMIN COMMAND 'perfmon'` command.

“Dynamically changing database cache size”
You can increase the database cache dynamically by using the **ADMIN COMMAND 'parameter'** command.

Dynamically changing database cache size

You can increase the database cache dynamically by using the **ADMIN COMMAND 'parameter'** command.

To increase the value of the **IndexFile.CacheSize** parameter dynamically, issue the following command:

```
ADMIN COMMAND 'parameter IndexFile.CacheSize=size'
```

The size unit is bytes. You can also specify the amount of space in units of megabytes, for example, "10M" for 10 megabytes.

Important: The cache size cannot be decreased dynamically. To decrease the cache size, edit the parameter value in the `solid.ini` configuration file and restart the server.

The solidDB server 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, which results in a higher collision probability. To avoid collision, use the **IndexFile.ReferenceCacheSizeForHash** parameter to accommodate the enlarged cache. The **IndexFile.ReferenceCacheSizeForHash** parameter value is used for calculating the cache hash table size. Use the parameter if you know the maximum cache size during the server lifecycle in advance. If the value is not given, hash table collisions might occur when the cache size is increased.

Note: The **ReferenceCacheSizeForHash** parameter value must not be smaller than the **IndexFile.CacheSize** value. If it is, the **IndexFile.ReferenceCacheSizeForHash** parameter value is rejected and the default value is used. Also, a message is printed to the `solmsg.out` log file.

Example

```
ADMIN COMMAND 'parameter IndexFile.CacheSize=40M'
```

8.3.4 Sorting

When the solidDB SQL Optimizer chooses an execution plan, it considers the performance impact of sorting data. Sorting occurs if the result set is not returned automatically in the correct order. If sorting is needed, the Optimizer chooses whether to use the internal sorter or the external sorter. The internal sorter is used with small result sets (hundreds of rows) while the external sorter is used with large result sets (thousands of rows).

Sorting occurs when no index satisfies the requested ordering of fetched rows. If the table data is accessed using the primary key or index, the result set is automatically in the order specified by the index in use. Hence, you can improve server performance by designing primary keys and indexes to support the ordering requirements of frequently used, performance-critical queries.

Note: Some queries require sorting implicitly. For example, if the optimizer chooses a JOIN operation to use the MERGE JOIN algorithm, the result sets to be joined require sorting before the join can occur.

Internal sorter

The internal sorter performs all sorting in the main memory. The amount of memory used for sorting is defined with the **SQL.SortArraySize** parameter. The **SQL.SortArraySize** parameter defines the size of the array (in rows) that is used for ordering the result set of a query. For example, if you specify a value of 1000, the server creates an array large enough to sort 1000 rows of data. If the amount of data to be sorted does not fit into the allocated memory, increase the value of the parameter **SQL.SortArraySize**.

External sorter

If the sorting task does not fit in the main memory (typically with large result sets), the Optimizer uses the external sorter, which stores intermediate information to disk. The external sorter is enabled by default (**Sorter.SorterEnabled=yes**).

The temporary files used by the external sort are created in a directory or directories specified with the **Sorter.TmpDir_N** parameter. The files are deleted automatically after sorting has finished.

To achieve better performance, the external sort files can be stored to a local drive using local disk names. Using multiple local disks avoids network I/O and balances the I/O load to multiple disks.

For example:

```
[Sorter]
TmpDir_1 = c:\tmp
TmpDir_2 = d:\tmp
TmpDir_3 = e:\tmp
```

An external sort requires space both on disk and in memory, not just space on the disk. You can configure the maximum amount of memory used for sorting with the **Sorter.MaxMemPerSort** and **Sorter.MaxCacheUsePercent** parameters.

Querying and controlling Optimizers sorter decisions

You can query the Optimizer decisions for sorting using the *EXPLAIN PLAN FOR* statement.

If the Optimizer is not choosing the optimal query execution plan, you can override the Optimizer decision by using optimizer hints. For more information, see *Hints* in the *IBM solidDB Programmer Guide*.

Additionally, the performance counters with the prefix *Sorter* provide information about the external sorter tasks. To view the Sorter performance counters, issue the following command:

```
ADMIN COMMAND 'pmon sorter'
```

For example, high values of the *Sorter start sort* counter indicate excessive use of the external sorter. If you have enough memory available, you can increase the value of the **SQL.SortArraySize** parameter to avoid the use of the external sorter.

8.4 Cache segment partitioning

To improve performance, you can partition the database cache (buffer pool) into segments that are cached independently. Specific tables or ranges of tables can be assigned to use a specific portion of the cache. When an SQL statement accesses a portion of table that is assigned to a specific cache segment, all operations in that query are cached into the same cache segment.

Using cache segment partitioning can be beneficial in setups where you can classify data into parts that do not interfere with each other. For example, if your application accumulates a lot of history-type data that is queried occasionally, you might not want the occasional queries to flush all cache pages so that concurrent active queries are slowed down. If you assign the history-type data to dedicated cache segments, you can limit the maximum cache size that the occasional queries can use. Active queries can thus have sufficient cache capacity. When the occasional reports are not being run, the segmented cache parts are released for use by the active queries.

Principles of operation

With cache segment partitioning, you can divide the database file cache into multiple segments. By default, no cache segments are defined. You create and modify the cache segments using SQL statements.

Each cache segment must have a unique name. The size of the segment is specified as a percentage of the total cache size. The total size for the assigned segments cannot be more than 80 % of the total cache size. The total cache size is defined with the **IndexFile.CacheSize** parameter.

You can assign the entire table or a part of a table to a cache segment. A part of a table is assigned to a cache segment by giving a column range. A single table can belong to only one cache segment, but a single cache segment can have multiple tables.

When an SQL statement accesses data that is associated to a cache segment, the entire SQL statement is assigned to that cache segment. All database caching for the SQL statement goes through that cache segment. If the SQL statement later accesses data from some other cache segment, it does not change the cache segment used for the statement.

From the application perspective, you do not need make any application level or SQL statement level changes to use cache segment partitioning. If cache pages are used for other purposes such as sorting or query buffering, the number of pages allocated to cache segments is adjusted accordingly. The system tries to keep the percentage of the cache allocated to a segment correct, even when the available size of the cache changes.

If there are no SQL statements using a cache segment, the cache pages can be used for the (unassigned) default segment, which ensures that cache pages are always fully utilized.

Using cache segment partitioning does not change the physical data representation or storage. The data association to a cache segment works only at a logical level based on column values. If the cache segments are changed (or removed), the new segments are taken into use after a server restart. If you do not restart the server, the cache segment usage is changed gradually.

Creating and modifying cache segments

To create cache segments:

1. Use the CREATE CACHE SEGMENT statement to define the name and size of the segment.
2. Use the ALTER TABLE ... ADD CACHE SEGMENT statement to assign a table or a data range to the segment.

   In addition to assigning entire tables, the data range can be defined using a WHERE clause of the following types:

```
[WHERE range_specification]

range_specification ::= <column> < <value>
                        <column> > <value>
                        <column> = <value>
                        <column> <> <value>
                        <column> BETWEEN <value> AND <value>
                        <column> OLDERTHAN <value> DAYS
                        <column> OLDERTHAN <value> SECONDS
                        <column> NEWERTHAN <value> DAYS
                        <column> NEWERTHAN <value> SECONDS
```

   The OLDERTHAN and NEWERTHAN syntax specifies date and time ranges; the data type of the columns must be DATE or TIMESTAMP.

To change the segment size:

1. Use the ALTER TABLE ... SET POOLSIZE statement to set the new segment size.

2. Restart the server to make the change effective immediately. If you do not restart the server, the change is gradual.

To remove cache segments:

1. Use the ALTER TABLE ... DROP CACHE SEGMENT statement to unassign a table from the segment.
2. Use the DROP CACHE SEGMENT statement to delete the cache segment definition from the database.
3. Restart the server to make the change effective immediately. If you do not restart the server, the change is gradual.

Example: Assigning tables to cache segments

To assign the table HISTORY_DATA and all of its contents to the cache segment HISTORY CACHE, use the following statements:

```
CREATE CACHE SEGMENT HISTORY CACHE
ALTER TABLE HISTORY_DATA ADD CACHE SEGMENT HISTORY_CACHE
```

Example: Assigning data ranges to cache segments

The table HISTORY_DATA contains a column DT of the data type DATE. To assign data that is older than seven days to the cache segment HISTORY CACHE, use the following statements:

```
CREATE CACHE SEGMENT HISTORY CACHE
ALTER TABLE HISTORY_DATA ADD CACHE SEGMENT HISTORY_CACHE WHERE DT OLDERTHAN 7 DAYS
```

Querying cache segment sizes and assignments

The cache segments and table assignments are stored in the system tables SYS_CACHESEGMENTS and SYS_CACHESEGMENT_CONDITIONS. Use the following type of statements to query the cache segment information.

To list tables that use a given cache segment:

```
SELECT TABLE_NAME, CACHESEGMENT_NAME from SYS_TABLES, SYS_CACHESEGMENT_CONDITIONS
where ID = BASE_TABLE_ID
  and CACHESEGMENT_NAME = 'segment_name'
```

For example:

```
SELECT TABLE_NAME, CACHESEGMENT_NAME from SYS_TABLES, SYS_CACHESEGMENT_CONDITIONS
where ID = BASE_TABLE_ID
  and CACHESEGMENT_NAME = 'CS1'

TABLE_NAME         CACHESEGMENT_NAME
----------         -----------------
TAB_TIMESTAMP      CS1
TAB_INTEGER        CS1
2 rows fetched.
```

To list cache segments that are used by a given table:

```
SELECT TABLE_NAME, CACHESEGMENT_NAME from SYS_CACHESEGMENT_CONDITIONS, SYS_TABLES
WHERE SYS_TABLES.ID = SYS_CACHESEGMENT_CONDITIONS.BASE_TABLE_ID
      and SYS_TABLES.TABLE_NAME = 'table_name'
```

For example:

```
SELECT TABLE_NAME, CACHESEGMENT_NAME from SYS_CACHESEGMENT_CONDITIONS, SYS_TABLES
WHERE SYS_TABLES.ID = SYS_CACHESEGMENT_CONDITIONS.BASE_TABLE_ID
      and SYS_TABLES.TABLE_NAME = 'TAB_TIMESTAMP'

TABLE_NAME         CACHESEGMENT_NAME
----------         -----------------
TAB_TIMESTAMP      CS1
1 rows fetched.
```

8.5 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, modify the following parameters:

- **Srv.RowsPerMessage**: The default value is 10.
- **Srv.ExecRowsPerMessage**: The default value is 2.

8.6 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.

8.6.1 Distributing I/O

Disk contention occurs when multiple processes try to access the same disk simultaneously. To avoid contention, 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. See "Managing database files and caching (IndexFile section)" on page 47 for more details.
- Consider using a separate disk for the external sorter

Typically it is 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 might want to run defragmentation software if one is available on your system. If your database file is growing, you might be able to reduce future file fragmentation by using the configuration parameter **IndexFile.ExtendIncrement**. Increasing the size of this parameter tells the server to allocate larger amounts of disk space when it runs out of space. Increasing the value does not guarantee contiguity because the operating system itself can allocate non-contiguous sectors to satisfy even a single request for more space. As a general rule, larger values of **IndexFile.ExtendIncrement** improve performance slightly, while smaller values keep the database size slightly smaller.

8.6.2 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 cause the merge process to start. The merge interval is controlled with the **General.MergeInterval** parameter. 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 (that is, 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 fills the available memory and the diskless server runs out of memory.

Note: Although the server has higher performance if merge intervals are less frequent (that is, batch inserts are larger), you might also see less consistent response times. If your highest priority is not overall throughput but to minimize the longest response time instead, you might want to make merge intervals more frequent rather than less frequent. More frequent merges reduce the worst case delays that interactive users can experience.

For details on detecting and preventing performance problems associated with Bonsai Tree growth, read 8.8, "Reducing Bonsai Tree size by committing transactions," on page 177.

8.7 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 runtime performance. Typically the performance impact of checkpoints is small. Similar to merge intervals, less frequent checkpoints can 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.

- Set configuration parameters in the `solid.ini` file.
  - Set the **`General.CheckpointInterval`** parameter. The default checkpoint interval is every 50000 log writes.
  - Set the **`General.MinCheckpointTime`** parameter.
- Force a checkpoint by using the `ADMIN COMMAND 'makecp'` command.

Frequent checkpoints can reduce the recovery time in the event of a system failure. If the checkpoint interval is small, 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; however, 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 takes. The database cache size is controlled with the **IndexFile.CacheSize** parameter.

Related reference:

A.4, “General section,” on page 217

Related information:

2.8, “Creating checkpoints,” on page 33

“CacheSize parameter” on page 49
The **IndexFile.CacheSize** parameter defines the amount of main memory that is used to maintain the shared buffer pool of a disk database. This buffer pool is called the database cache.

8.1, “Logging and transaction durability,” on page 163

8.8 Reducing Bonsai Tree size by committing transactions

The solidDB server provides a consistent view of data within one transaction. If a user does not commit a transaction, the server 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 Bonsai Tree functionality, 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, the server requires more memory. If the amount of memory available is insufficient, the server 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 tools and tools provided with thesolidDB server.

8.8.1 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.)

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 41. Determining command status

Command/File	Information
ADMIN COMMAND 'userlist'	Obtain a list of existing connections.
ADMIN COMMAND 'status'	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 42. 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, the statement has not closed or committed during this interval.

Providing COMMIT statements in the application code

To make sure that every database operation gets committed, perform one of the following operations:

- Execute the statement COMMIT WORK.
- Call ODBC function SQLTransact or SQLEndTran.
- Call JDBC method commit.

Make sure that 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 receives only the return code 'SUCCESS' from the ODBC Driver Manager, even though no transaction is committed in the database. This situation can go unnoticed. In addition to the ODBC Driver Manager or solidDB SQL Editor (**solsql**), other utilities can also have open transactions.

Make sure that you are aware of all database connections. Each FETCH after COMMIT (keeping the statement handle alive) also causes a new transaction to start.

8.9 Diagnosing poor performance

There are different areas in solidDB that can result in performance degradation. In order to remedy performance problems, you need to determine the underlying cause. Following is a table that lists common symptoms of poor performance, possible causes, and directs you to the section in this chapter for the remedy.

Table 43. 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 *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, see *EXPLAIN PLAN FOR statement*. If the Optimizer is not choosing the optimal query execution plan, override the Optimizer decision by using optimizer hints. For more details, see *Using optimizer hints*.
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 0.5 MB per concurrent user or 2-5% of the database size. For more details, read the section *Defining database cache size* in *IBM solidDB Administrator Guide*.
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* in *IBM solidDB Administrator Guide*.

Table 43. Diagnosing poor performance (continued)

Symptoms	Diagnosis	Solution
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 *Optimizing batch inserts and update*.
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.

9 Troubleshooting and support

To help you understand, isolate, and resolve problems with your solidDB products, the troubleshooting and support information contains instructions for using the problem-determination resources that are provided with your solidDB products.

To resolve a problem on your own, you can find out how to identify the source of a problem, how to gather diagnostic information, where to get fixes, and which knowledge bases to search. If you need to contact IBM Software Support, you can find out what diagnostic information the service technicians require to help you address a problem.

9.1 Troubleshooting a problem

Troubleshooting is a systematic approach to solving a problem. The goal of troubleshooting is to determine why something does not work as expected and how to resolve the problem.

The first step in the troubleshooting process is to describe the problem completely. Problem descriptions help you and your IBM Support representative know where to start to find the cause of the problem. This step includes asking yourself basic questions:

- What are the symptoms of the problem?
- Where does the problem occur?
- When does the problem occur?
- Under which conditions does the problem occur?
- Can the problem be reproduced?

The answers to these questions typically lead to a good description of the problem, which can then lead you a problem resolution.

What are the symptoms of the problem?

When starting to describe a problem, the most obvious question is "What is the problem?" This question might seem straightforward; however, you can break it down into several more-focused questions that create a more descriptive picture of the problem. These questions can include:

- Who or what is reporting the problem?
- What are the error codes and messages?
- How does the system fail? For example, is it a loop, hang, crash, performance degradation, or incorrect result?

Where does the problem occur?

Determining where the problem originates is not always easy, but it is one of the most important steps in resolving a problem. Many layers of technology can exist between the reporting and failing components. Networks, disks, and drivers are only a few of the components to consider when you are investigating problems.

The following questions help you to focus on where the problem occurs to isolate the problem layer:

- Is the problem specific to one platform or operating system, or is it common across multiple platforms or operating systems?
- Is the current environment and configuration supported?
- Is the application running locally on the database server or on a remote server?

If one layer reports the problem, the problem does not necessarily originate in that layer. Part of identifying where a problem originates is understanding the environment in which it exists. Take some time to completely describe the problem environment, including the operating system and version, all corresponding software and versions, and hardware information. Confirm that you are running within an environment that is a supported configuration; many problems can be traced back to incompatible levels of software that are not intended to run together or have not been fully tested together.

When does the problem occur?

Develop a detailed timeline of events leading up to a failure, especially for those cases that are one-time occurrences. You can most easily develop a timeline by working backward: Start at the time an error was reported (as precisely as possible, even down to the millisecond), and work backward through the available logs and information. Typically, you need to look only as far as the first suspicious event that you find in a diagnostic log.

To develop a detailed timeline of events, answer these questions:

- Does the problem occur only at a certain time of day or night?
- How often does the problem occur?
- What sequence of events leads up to the time that the problem is reported?
- Does the problem occur after an environment change, such as upgrading or installing software or hardware?

Responding to these types of questions can give you a frame of reference in which to investigate the problem.

Under which conditions does the problem occur?

Knowing which systems and applications are running at the time that a problem occurs is an important part of troubleshooting. These questions about your environment can help you to identify the root cause of the problem:

- Does the problem always occur when the same task is being performed?
- Does a certain sequence of events need to occur for the problem to surface?
- Do any other applications fail at the same time?

Answering these types of questions can help you explain the environment in which the problem occurs and correlate any dependencies. Remember that just because multiple problems might have occurred around the same time, the problems are not necessarily related.

Can the problem be reproduced?

From a troubleshooting standpoint, the ideal problem is one that can be reproduced. Typically, when a problem can be reproduced you have a larger set of tools or procedures at your disposal to help you investigate. Consequently, problems that you can reproduce are often easier to debug and solve. However, problems that you can reproduce can have a disadvantage: If the problem is of

significant business impact, you do not want it to recur. If possible, re-create the problem in a test or development environment, which typically offers you more flexibility and control during your investigation

- Can the problem be re-created on a test system?
- Are multiple users or applications encountering the same type of problem?
- Can the problem be re-created by running a single command, a set of commands, or a particular application?

9.1.1 Tools for troubleshooting

The following tools are available to help collect, format, or analyze diagnostic data.

- **ADMIN COMMAND 'userlist'**

  The **ADMIN COMMAND 'userlist -l'** command displays a list of users currently logged in to the database. The output provides information about various database operations and settings for each user.

- **ADMIN COMMAND 'report'**

  The **ADMIN COMMAND 'report'** command produces a report that contains information about the server, users, and database operations. The report also includes the configuration file (`solid.ini`) settings and a list of the performance counters.

- **ADMIN COMMAND 'pmon'**

  The **ADMIN COMMAND 'pmon'** command displays the solidDB performance counters (called *perfmons* or *pmons*) that provide information about various database operations and performance

- **ADMIN COMMAND 'status'**

  The **ADMIN COMMAND 'status'** command displays statistics information about memory usage, process size, transaction count, cache count, user count, database operations.

- **ADMIN COMMAND 'monitor'**

  The **ADMIN COMMAND 'monitor'** command controls monitoring of user activity and SQL calls. The information is logged into the `soltrace.out` file. Monitoring can also be turned on with the command-line option **-m** at solidDB startup.

- **ADMIN COMMAND 'trace'**

  The **ADMIN COMMAND 'trace'** command controls the solidDB trace facility.

- **ADMIN COMMAND 'sqllist'**

  The **ADMIN COMMAND 'sqllist'** command displays a list of the longest running SQL statements among the currently running statements. You can limit the number of statements shown by specifying the number of statements as an attribute (**ADMIN COMMAND 'sqllist top <no_of_statements>'**).

- **ADMIN COMMAND 'backuplist'**

  The **ADMIN COMMAND 'backuplist'** command displays the status of the last local backup.

- **ADMIN COMMAND 'proctrace'**

  The **ADMIN COMMAND 'proctrace'** command controls tracing in stored procedures and triggers.

- **EXPLAIN PLAN FOR**

  The **EXPLAIN PLAN FOR** SQL statement shows the execution plan that the SQL optimizer has selected for a given SQL statement.

- ODBC Driver Manager trace facility (Windows)

The Windows ODBC Driver Manager has a trace facility that allows the sequence of function calls made by an ODBC application to be recorded into a log file.

Tracing SQL statements

You can trace SQL statements using the ADMIN COMMAND 'trace' and ADMIN COMMAND 'monitor' commands or by using the SQL Info facility.

ADMIN COMMAND 'trace'

The **ADMIN COMMAND 'trace'** command controls the solidDB trace facility. The **ADMIN COMMAND 'trace on sql'** enables tracing of SQL statements. The tracing information is output by default to the `soltrace.out` file.

ADMIN COMMAND 'monitor'

The **ADMIN COMMAND 'monitor'** command controls the solidDB monitoring facility. The **ADMIN COMMAND 'monitor on'** enables monitoring of user activity and SQL calls. The monitoring logs are output to the `soltrace.out` file.

SQL Info facility

The SQL Info facility generates information for each SQL statement processed by solidDB.

To generate the SQL Info, you run your application with the SQL Info facility enabled. The SQL Info facility can be enabled in the following ways:

- **Sql.Info=<info_level>** parameter
- **ADMIN COMMAND 'trace on info <info_level>'** command
- `SET SQL INFO ON LEVEL` *`info_level`* `FILE` *`file_name`* statement

The tracing level (*info_level*) is defined as an integer between 0 (no tracing) and 8 (solidDB info from every fetched row).

Table 44. SQL Info levels

Info level	Description
0	no output
1	table, index, and view info in SQL format
2	SQL execution graphs (technical support use only)
3	some SQL estimate info, solidDB selected key name
4	all SQL estimate info, solidDB selected key info
5	solidDB info also from discarded keys
6	solidDB table level info
7	SQL info from every fetched row
8	solidDB info from every fetched row

The trace information is output by default to the `soltrace.out` file in the solidDB working directory. You can also specify the output file using the **SQL.InfoFileName** parameter. This is recommended since the `soltrace.out` file may contain information from several sources.

Examples

```
[SQL]
Info = 1
InfoFileName = solidsql_trace.txt
```

The following command turns on the SQL Info facility on level 3, outputting the trace information to a my_query.txt file in the working directory. This SQL Info facility is turned on only for the client that executes the statement.

```
SET SQL INFO ON LEVEL 1 FILE 'my_query.txt'
```

The following SQL statement turns off the SQL Info facility:

```
SET SQL INFO OFF
```

Analyzing Monitor facility trace output:

The Monitor facility provides information that you can use to analyze and tune performance at the individual SQL statement level.

The following example shows the contents of the Monitor output in `soltrace.out` for a solidDB server that is running a simplified transaction from a benchmark. In the example, the following statements are issued:

- SET
- SELECT (2 statements)
- UPDATE
- INSERT

```
------------------------------------------------------------
2012-11-17 09:21:22
Version: 7.0.0.0 Build 2012-10-03
Operating system: Linux 2.6.18 AMD64 64bit MT IBM solidDB 7.0
2012-11-17 09:21:23 User 'DBA' connected, user id 23, machine id
coralxib02.torolab.ibm.com (127.0.0.1).
2012-11-17 09:21:23.676 23:0:opencursor SQL_CUR1 'SET PASSTHROUGH READ NONE
WRITE NONE'
2012-11-17 09:21:23.676 23:0:execute SET PASSTHROUGH READ NONE WRITE NONE
2012-11-17 09:21:23.676 23:0:exec rowcount 0
2012-11-17 09:21:23.677 23:1:opencursor SQL_CUR2 'Select C_LAST, C_CREDIT,
C_DISCOUNT, W_TAX from CUSTOMER, WAREHOUSE where C_W_ID = ? and C_D_ID = ? and
C_ID = ? and W_ID = ?'
2012-11-17 09:21:23.677 23:2:opencursor SQL_CUR3 'Select D_NEXT_O_ID, D_TAX
from DISTRICT where D_W_ID = ? and D_ID = ? for update'
2012-11-17 09:21:23.678 23:3:opencursor SQL_CUR4 'Update DISTRICT set
D_NEXT_O_ID = ? where D_W_ID = ? and D_ID = ?'
2012-11-17 09:21:23.678 23:4:opencursor SQL_CUR5 'Insert into ORDERS values (?,
?, ?, ?, ?, ?, ?, ?)'
2012-11-17 09:21:23.678 23:1:execute Select C_LAST, C_CREDIT, C_DISCOUNT, W_TAX
from CUSTOMER, WAREHOUSE where C_W_ID = ? and C_D_ID = ? and C_ID = ? and W_ID
= ?
2012-11-17 09:21:23.678 23:1:param 1:3838
2012-11-17 09:21:23.678 23:1:param 2:2
2012-11-17 09:21:23.678 23:1:param 3:23
2012-11-17 09:21:23.678 23:1:param 4:3838
2012-11-17 09:21:23.679 23:1:fetch next, 1 rows, total 1
2012-11-17 09:21:23.679 23:2:execute Select D_NEXT_O_ID, D_TAX from DISTRICT
where D_W_ID = ? and D_ID = ? for update
2012-11-17 09:21:23.679 23:2:param 1:3838
2012-11-17 09:21:23.679 23:2:param 2:2
2012-11-17 09:21:23.679 23:2:fetch next, 1 rows, total 1
2012-11-17 09:21:23.679 23:3:execute Update DISTRICT set D_NEXT_O_ID = ? where
D_W_ID = ? and D_ID = ?
2012-11-17 09:21:23.679 23:3:param 1:32
2012-11-17 09:21:23.679 23:3:param 2:3838
2012-11-17 09:21:23.679 23:3:param 3:2
2012-11-17 09:21:23.679 23:3:exec rowcount 1
2012-11-17 09:21:23.679 23:4:execute Insert into ORDERS values (?, ?, ?, ?, ?,
?, ?, ?)
```

```
2012-11-17 09:21:23.680 23:4:param 1:31
2012-11-17 09:21:23.680 23:4:param 2:23
2012-11-17 09:21:23.680 23:4:param 3:2
2012-11-17 09:21:23.680 23:4:param 4:3838
2012-11-17 09:21:23.680 23:4:param 5:2012-11-17 09:21:23
2012-11-17 09:21:23.680 23:4:param 6:NULL
2012-11-17 09:21:23.680 23:4:param 7:8
2012-11-17 09:21:23.680 23:4:param 8:1
2012-11-17 09:21:23.680 23:4:exec rowcount 1
2012-11-17 09:21:23.680 23:transopt commit (6)
2012-11-17 09:21:23.680 23:0:close
2012-11-17 09:21:23.680 23:1:close
2012-11-17 09:21:23.680 23:2:close
2012-11-17 09:21:23.681 23:3:close
2012-11-17 09:21:23.681 23:4:close
2012-11-17 09:21:23 User 'DBA' disconnected, user id 23, machine id
coralxib02.orolab.ibm.com (127.0.0.1).
```

The output can be divided into four columns as follows:

```
2012-11-17 09:21:23.679 | 23: | 3: | execute Update DISTRICT set D_NEXT_O_ID = ? where
D_W_ID = ? and D_ID = ?
```

Timestamp (2012-11-17 09:21:23.679)

The first column is the time stamp. To ensure that the output has millisecond precision, the **SRV.TraceSecDecimals** parameter must be set to 3 (default).

Connection ID (23)

The second column is the connection ID. This number identifies each client connection to the solidDB server uniquely. In the example, only one connection is used, which is represented by connection ID 23. The example also shows when the user connected and disconnected.

Statement ID or transaction operation (0-4)

The column token is either a statement ID or a transaction level operation. The example shows the output from five SQL statements that are run within one transaction. The statement IDs for the five statements vary from 0 to 4. When a workload is running with more than one client that runs many SQL statements, the combination of connection ID and statement ID can identify each entry in the trace output uniquely. When the third token is not the statement ID, it is usually a transaction level operation, such as commit or rollback.

You can use grep or search facilities in any file viewing utility to use the combination to isolate and view one sequence of operations quickly.

Tip: To limit the amount of output the server produces, you can enable the Monitor facility for a specific user only by issuing the command `ADMIN COMMAND 'monitor on user username'`.

Trace data (execute Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID = ? and D_ID = ?)

The fourth column shows the actual trace data for the operation. It can be the actual SQL statement being prepared or executed, the parameters being used, or another statement level operation that the server is performing.

Example: Analysing the Monitor facility output

SELECT statement (ID 2)

Focusing on statement ID 2, which is a SELECT statement, you can see that the start timestamp for the prepare, shown as `opencursor` followed by the internally assigned cursor identifier is 2010-11-17 09:21:23.677. The execute started at 2010-11-17 09:21:23.679, which means that prepare took about 2 milliseconds to complete.

The fetch completed at 2010-11-17 09:21:23.679, therefore it appears to have taken 0 milliseconds. The 0 millisecond duration typically means that the execution completed in sub-milliseconds or microseconds. Because the timer precision cannot be set to show microseconds, microsecond-level information is not available. Because of the precision limitation, the duration of the operations is approximate.

In the example output, the `prepare` operation takes about twice as long as the `execute` operation. This aligns with the known fact that preparing SQL statements is more expensive than executing them. To optimize the performance of your database, prepare statements as few times as possible.

In the example output, the statement ID 2 fetches a total of 1 row. Typically, the more rows you need to fetch, the longer the statement execution takes and the less advantage an in-memory database has over traditional disk-based database management systems.

INSERT statement (ID 4)

Focusing on statement ID 4, which is an INSERT statement, you can see that the `prepare` operation started at 2010-11-17 09:21:23.678, the `execute` started at 2010-11-17 09:21:23.679, and the `execute` completed at 2010-11-17 09:21:23.680. By looking at the time stamps, you can see the `prepare` took about 1 millisecond and the `execute` took less than 1 millisecond.

Transaction duration

You can use the timestamp information to calculate the duration of a transaction. In the example, the transaction executed by connection ID 23 started at about 2010-11-17 09:21:23.676. The end of the transaction execution is marked by the by the operation `transopt commit (6)` (the digit 6 in parentheses is an internal identifier for a commit transaction operation). The timestamp associated with `transopt commit (6)` is 2010-11-17 09:21:23.680. It took about 4 milliseconds to complete the transaction.

Reconstructing executed SQL statements

The trace output shows the parameter values for the dynamic SQL statements. You can use the trace output to reconstruct the actual SQL statements that were executed. For example, you can execute the statements with the same parameters in solidDB SQL Editor (**solsql**) to analyze the statement further.

Analyzing SQL Trace facility output:

The SQL Trace facility provides information that you can use to analyze and tune performance at the individual SQL statement level.

The following example shows the contents of the SQL Trace output in `soltrace.out` for a solidDB server that is running a simplified transaction from a benchmark. In the example, the following statements are issued:

- SET
- SELECT (2 statements)
- UPDATE
- INSERT

```
2012-11-17 11:11:38.959 2:sql:161:prepare SET PASSTHROUGH READ NONE WRITE NONE
2012-11-17 11:11:38.959 2:sql:161:execute:SET PASSTHROUGH READ NONE WRITE NONE
2012-11-17 11:11:38.960 2:sql:163:prepare SELECT C_LAST, C_CREDIT, C_DISCOUNT,
W_TAX FROM CUSTOMER, WAREHOUSE WHERE C_W_ID = ? AND C_D_ID = ? AND C_ID = ? AND
W_ID = ?
2012-11-17 11:11:38.961 2:sql:164:prepare SELECT D_NEXT_O_ID, D_TAX FROM
```

```
DISTRICT WHERE D_W_ID = ? AND D_ID = ? FOR UPDATE
2012-11-17 11:11:38.961 2:sql:165:prepare UPDATE DISTRICT SET D_NEXT_O_ID = ?
WHERE D_W_ID = ? AND D_ID = ?
2012-11-17 11:11:38.961 2:sql:166:prepare INSERT INTO ORDERS VALUES (?, ?, ?,
?, ?, ?, ?, ?)
2012-11-17 11:11:38.961 2:sql:trans begin
2012-11-17 11:11:38.961 2:sql:163:execute:SELECT C_LAST, C_CREDIT, C_DISCOUNT,
W_TAX FROM CUSTOMER, WAREHOUSE WHERE C_W_ID = ? AND C_D_ID = ? AND C_ID = ? AND
W_ID = ?
2012-11-17 11:11:38.962 2:sql:163:fetch
2012-11-17 11:11:38.962 2:sql:164:execute:SELECT D_NEXT_O_ID, D_TAX FROM
DISTRICT WHERE D_W_ID = ? AND D_ID = ? FOR UPDATE
2012-11-17 11:11:38.962 2:sql:164:fetch
2012-11-17 11:11:38.962 2:sql:165:execute:UPDATE DISTRICT SET D_NEXT_O_ID = ?
WHERE D_W_ID = ? AND D_ID = ?
2012-11-17 11:11:38.962 2:sql:stmt commit (0)
2012-11-17 11:11:38.963 2:sql:166:execute:INSERT INTO ORDERS VALUES (?, ?, ?,
?, ?, ?, ?, ?)
2012-11-17 11:11:38.963 2:sql:stmt commit (0)
2012-11-17 11:11:38.963 2:sql:trans commit (0)
2012-11-17 11:11:38.963 2:sql:161:close
2012-11-17 11:11:38.963 2:sql:163:close
2012-11-17 11:11:38.963 2:sql:164:close
2012-11-17 11:11:38.963 2:sql:165:close
2012-11-17 11:11:38.963 2:sql:166:close
```

The output can be divided into five columns as follows:

```
2012-11-17 11:11:38.962 | 2: | sql: |165: | execute:UPDATE DISTRICT SET D_NEXT_O_ID = ?
WHERE D_W_ID = ? AND D_ID = ?
```

Timestamp (2012-11-17 11:11:38.962)
The first column is the time stamp. To ensure that the output has millisecond precision, the **SRV.TraceSecDecimals** parameter must be set to 3 (default).

Connection ID (2)
The second column is the connection ID. This number identifies each client connection to the solidDB server uniquely. In the example, only one connection is used, which is represented by connection ID 23. The example also shows when the user connected and disconnected.

sql The third column is the SQL trace identifier sql. It is displayed so that you can differentiate the SQL trace information from other component trace information in the trace file.

Transaction ID or transaction operation (165)
The fourth column is either a transaction statement ID (such as 165) or a transaction level operation (such as `trans begin`). The transaction ID is an internal number assigned by the server to each transaction.

Note: The transaction ID in the SQL Trace output differs from the statement ID in the Monitor facility output.

The `trans begin` token is output when the transaction is started. In solidDB, transactions are started during the first SQL statement execution. Prepares and most SET statements are not part of a transaction.

The `trans commit` token is output when the transaction is committed. You can use the `trans begin` and `trans commit` tokens to calculate the duration of the transaction. For example, the timestamp for the `trans begin` token is 2010-11-17 11:11:38.961 and the timestamp for the `trans commit` token is 2010-11-17 11:11:38.963. Therefore, this transaction took approximately 2 milliseconds to complete.

The value in parentheses after the `trans commit` token is the return code of the commit.

Trace data (execute:UPDATE DISTRICT SET D_NEXT_O_ID = ? WHERE D_W_ID = ? AND D_ID = ?)
The fifth column shows the actual trace data for the operation. It can be

the actual SQL statement being prepared or executed, the parameters being used, or another statement level operation that the server is performing.

The SQL Trace facility does not include the dynamic SQL parameter values in the output. You cannot use the SQL Trace output to reconstruct exact SQL execution as you can with the Monitor facility. The SQL Trace facility is best for analyzing the flow of statement execution.

Comparison of the Monitor facility and the SQL Trace facility:

The output of the Monitor facility and the SQL Trace facility is slightly different. For example, you need to use the SQL Trace facility to trace statements that are executed in stored procedures.

Table 45. Comparison of the Monitor facility and the SQL Trace facility

Description	Monitor	SQL Trace
Trace SQL statements executed in stored procedures	No	Yes
Dynamic SQL parameter values	Yes	No
Statement row counts	Yes	No
Commit return code	No	Yes
User connect and disconnect messages	Yes	No
trans begin displayed at the start of transactions	No	Yes
Same statement ID displayed as in **ADMIN COMMAND 'userlist'** and **ADMIN COMMAND 'sqllist'** outputs	Yes	No

Using stack trace facility

The stack traces facility collects diagnostics information upon server failures. In general, IBM Software Support and development teams use the stack traces facility for troubleshooting. You can also generate stack traces to gain information about a problem that you are investigating, but its use is rather limited without knowledge of the solidDB source code.

About this task

The stack traces facility is controlled with the **Srv.StackTraceEnabled** parameter. When set to 'yes' (default), the stack trace information is output to `ssstacktrace-<process_id>-<thread_id>.out` file in the solidDB working directory.

The following signals invoke the stack traces output automatically:

- SIGSEGV
- SIGILL
- SIGBUS
- SIGTRAP
- SIGSYS
- SIGEMT

The stack traces information is produced only about the thread that received the signal.

Additionally, you can generate the stack traces information for all currently running threads by sending the server the SIGUSR1 signal.

Note: The stack traces facility is not supported on Windows operating systems.

Procedure

- To enable or disable the stack traces facility, set the **Srv.StackTraceEnabled** parameter to 'yes' or 'no'.
- To output the stack trace information manually without shutting down the server, send the server the SIGUSR1 signal.

  For example, use the following command in Linux environments:

  ```
  kill -SIGUSR1 <process_id>
  ```

Tracing communication between client and server

solidDB provides the following tools for observing the communication between an application and a database server:

- Network trace facility

  Use the network trace facility when you want to know why a connection is not established to the solidDB server.
- Ping facility

  Use the ping facility to determine how fast packets are transferred between an application and the solidDB server.

Network trace facility: Network tracing can be done on the solidDB node, on the application node, or concurrently on both nodes. The trace information is written to the default trace file or the file specified with the **Com.TraceFile** parameter.

The default name of the output file is `soltrace.out`. This file is created in 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

You can turn on the network trace facility in the following ways:

- Use the **Com.Trace** and **Com.TraceFile** parameters.

  Defining the **TraceFile** configuration parameter automatically turns on the Network trace facility.
- Use the environment variables SOLTRACE and SOLTRACEFILE.

  The environment variable settings override the definitions in the `solid.ini` file.

  Defining the SOLTRACEFILE environment variable automatically turns on the Network trace facility.
- Use the option `-t` and/or `-ofilename` as a part of the network name.
  - Option `-t` turns on the Network trace facility.
  - Option `-o` turns on the facility and defines the name of the trace output file.

Defining trace parameters in the client-side configuration file

```
[Com]
Trace ={Yes|No}
; default No
TraceFile = file_name
; default soltrace.out
```

For example:

```
[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'.
```

Ping facility: The solidDB ping facility can be used to test the performance and functionality of the network connection. The ping facility is built into all solidDB client applications and is turned on with the network name option **-p** *level* .

The output file is 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 can be used only if the server is set to use the ping facility at least at the same level.

Table 46. 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 the solidDB client does not have an existing server connection, you can use the SQLConnect() function with the connect string option **-p1** (ping test, level 1) to check if solidDB is listening in a certain address. Without logging in to 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

Turn on the ping facility by using the following network name syntax:

```
protocol_name -p level server_name
```

For example, to run the ping facility with solidDB SQL Editor (**solsql**), use the following command:

```
solsql "tcp -p1 -oping.out 1964"
```

The above command runs the ping facility at the level 1 and outputs the results into `soltrace.out`. The ping facility 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
```

Com.Listen parameter and restrictions on the ping facility

The server-side ping level that is set with the **Com.Listen** parameter restricts the available ping levels on the client side. Clients can always use the ping facility at level 1 (0 is no operation/default). Levels 2, 3, 4 or 5 can be used only if the server is set to use the ping facility at least at the same level.

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 SQL clients connected to the same solidDB.

9.1.2 Troubleshooting licensing issues

The solidDB package includes an evaluation license file (`solideval.lic`) that you can use to evaluate the solidDB product for a limited time. After the evaluation license expires, you must install the full license to continue to use the database.

Evaluation license has expired

Symptom

After the evaluation license has expired, you receive the following type of error and you cannot start the server:

```
SOLID System Fatal Error 11014: Database age limit of evaluation license expired.
SOLID System Fatal Error 11015: Evaluation license expired.
```

Resolving the problem

To continue to use your database after the evaluation license has expired, you must install the full license on your environment.

The `solid.lic` license file is distributed as a separate download image called the License Certificate. The solidDB License Certificate is available for download at IBM Passport Advantage®. In physical media deliveries, the License Certificate is included in the Quick Start DVD.

After you have downloaded the license certificate:

1. Unarchive the download image.
2. Copy the `solid.lic` file to your solidDB working directory or the location that is defined with the SOLIDDIR environment variable.

Tip: When you are using the evaluation license, the number of days that are left in your grace period is printed to the `solmsg.out` file each time the solidDB server starts.

For example:

```
5 more days to evaluate IBM solidDB.
To continue using this database without
disruption, see ORDERME.TXT.
```

License is not valid for this server version

Symptom

The server startup fails. The following message is output to `solerror.out`.

```
SOLID System Fatal Error 11019: License is not valid for this server version.
Exiting the program
```

Causes

The license files are not compatible between different server versions. When you upgrade your environment to a new solidDB server version level, you must update also the license file. Fix packs do not require a license file update.

Resolving the problem

1. Check the `solmsg.out` message log file for the location of the license file that the server is using. The license file location is output at the server startup. For example:

   ```
   2012-12-12 12:32:37
   Version: 7.0.0.3 Build 2012-11-06
   Operating system: Windows 64bit MT
   IBM solidDB - Version 7.0.0.3 Build 2012-11-06 (Windows 64bit MT)
   Copyright Oy International Business Machines Ab 1993, 2012.
   Strong encryption disabled, using default.
   Using license file C:\Program Files\IBM\solidDB\solidDB7.0\testdb\solid.lic
   ```

2. Update the license file.

   Note: If your 6.5 or 6.3 installation used the `soliduc.lic` license file, you must remove the `soliduc.lic` file and replace it with the `solid.lic` license file provided in the V7.0 License Certificate image.

License information not found

Symptom

The server startup fails. The following message is output to `solerror.out`.

```
SOLID System Fatal Error 11012: License information not found.
Exiting the program
```

Resolving the problem

Check that license file is available in your solidDB working directory or the location that is defined with the SOLIDDIR environment variable.

Related concepts:

2.3.3, “Setting up database environment,” on page 16
By default the solidDB database files, log, message, and trace files are created in the solidDB working directory. For production environments, you might want to set up an environment where, for example, database files, backup files, and log files are located on different disks.

9.1.3 Troubleshooting Universal Cache

This section provides instructions and guidelines on how to prevent or troubleshoot common problems while configuring or using Universal Cache.

• “Initial connections are not successful” on page 195

- "Dependencies between components used in replication"
- "Making changes to replication subscriptions"
- "Subscriptions fail after performing hsb netcopy followed by a switchover"
- "InfoSphere CDC for solidDB connection to solidDB server times out" on page 196
- "Bidirectional replication does not work between solidDB and DB2" on page 197

Initial connections are not successful

The components for Universal Cache must be installed and configured in the order described in section *Overview of installation and configuration steps*. Review the steps below and ensure that the installation and configuration steps were followed.

Installation and configuration order

- Frontend solidDB server
- InfoSphere CDC for solidDB
- Backend data server
- InfoSphere CDC for the backend data server
- Access Server
- Management Console

Dependencies between components used in replication

To set up replication between databases, you need define and create various entities and components which are dependent on each other. These entities and components must be created in the following order and modified or deleted in the reverse order. For more details and instructions, see the IBM InfoSphere Change Data Capture version 6.5 Information Center.

1. Databases
2. InfoSphere CDC instances
3. Datastores
4. Subscriptions
5. Table mappings

Making changes to replication subscriptions

If you need to make changes to your replication subscriptions, you must first end replication on your subscriptions. For more details and instructions, see section *Ending replication on a subscription* in the IBM InfoSphere Change Data Capture version 6.5 Information Center.

Subscriptions fail after performing `hsb netcopy` followed by a switchover

In High Availability (HotStandby) configurations, subscriptions where the solidDB database is the source datastore might fail if a switchover is performed shortly after **`hsb netcopy`**.

The subscriptions might fail, for example, in the following cases:

1. After a failure or a maintenance break, primary server (node 1) and secondary server (node 2) are synchronized using `ADMIN COMMAND 'hsb netcopy'`.
2. Replication continues against the primary server (node 1) for few transactions.

3. The primary server (node 1) fails and switchover changes the secondary server (node 2) to be the new primary server.
4. Subscriptions fail and replication against the new primary server (node 2) cannot be restarted.

Causes

The command `ADMIN COMMAND 'hsb netcopy'` does not copy any log files. Subsequently, because InfoSphere CDC replication is asynchronous in nature, InfoSphere CDC for solidDB might not have processed all the transactions up to the point from which the **hsb netcopy** was made. This means that the log position InfoSphere CDC for solidDB tries to use after the switchover might not be valid – the log entry for the last transaction on node 1 before the **hsb netcopy** might not exist on the new primary (node 2).

Workaround

To ensure that InfoSphere CDC for solidDB has access to a valid log entry in the new primary server (node 2) after a switchover:

- Before performing **hsb netcopy**, copy the log files from the primary server (node 1) to the secondary server (node 2). This ensures that InfoSphere CDC for solidDB has access to the log positions of the transactions that were executed before the **hsb netcopy** was made.

  or

- Do not perform switchover shortly after **hsb netcopy** or wait for several transactions to be replicated to the backend database before performing the switchover. This ensures that log positions in the primary server (node 1) and secondary server (node 2) are synchronized.

  or

- If the switchover has already taken place (for example, due to a failure of node 1):
  1. Recover the old primary server (node 1).
  2. Perform a switchover to return the old primary server (node 1) back to a primary server.
  3. Restart replication on the subscription.

  Before performing another switchover (to make node 2 the new primary server), wait for several transactions to be replicated. This ensures that log positions in the primary server (node 1) and secondary server (node 2) are synchronized.

InfoSphere CDC for solidDB connection to solidDB server times out

InfoSphere CDC for solidDB connections to the solidDB server can be idle for long periods of time, causing connection idle timeouts. By default, the solidDB server timeout for idle connections is set to 480 minutes (specified with the **Srv.ConnectTimeOut** parameter).

Workaround:

Set the connection idle timeout for the InfoSphere CDC for solidDB connection to infinite by using the non-standard solidDB JDBC connection property **solid_idle_timeout_min=0**. The InfoSphere CDC for solidDB connection settings are specified with the InfoSphere CDC configuration tool (**dmconfigurets**), using

the **Database area** > **Advanced** button in Windows operating systems or the **Configure advanced parameters** > **Modify settings** option in Linux and UNIX operating systems.

Note: The timeout setting specified for the InfoSphere CDC for solidDB instance does not impact the server setting (**Srv.ConnectTimeOut**) for other connections.

Bidirectional replication does not work between solidDB and DB2®

Symptom

Bidirectional replication between solidDB and DB2 for Linux, UNIX, and Windows does not work. You can create the subscriptions and table mappings successfully but data changes in the solidDB database are not replicated to the DB2 database.

In some cases, starting replication (**Start Mirroring**) fails with the following type of error message:

```
Error 1465 BIDI5 Mar 13, 2013 3:04:11 PM
--- Subscription BIDI5 is terminating abnormally.

Error 1713 BIDI5 Mar 13, 2013 3:04:11 PM
IBM InfoSphere Change Data Capture to BIDI5 is initiating shutdown due
to failure on the local system. See the previous messages for
additional information.

Error 2913 BIDI5 Mar 13, 2013 3:04:11 PM LOAD operation is not supported
by IBM InfoSphere Change Data Capture. Please refresh the the table [
DB2ADMIN.T42] when LOAD operation ends.
```

Recovery

To use bidirectional replication with DB2 for Linux, UNIX, and Windows, set the InfoSphere CDC for DB2 system parameter **ddl_awareness** to `false`.

You can set system parameters in two ways:

- On the computer where your InfoSphere CDC for DB2 replication engine is installed, issue the following command:

  ```
  dmset -I <INSTANCE_NAME> ddl_awareness=false
  ```

  For example:

  ```
  dmset -I backend_DB2 ddl_awareness=false
  ```

- In the **Configuration** perspective of the Management Console, right-click on the datastore and select **Properties** > **System Parameters**. Click **Add** and enter the parameter name and its value (**ddl_awareness=false**).

9.1.4 Troubleshooting SMA

This section provides instructions and guidelines on how to prevent or troubleshoot common problems while configuring or using SMA.

Error: Server could not allocate shared memory segment by id -1

Symptoms

When trying to start a SMA server, the following type of error is displayed, and the SMA server cannot be started.

```
IBM solidDB process has encountered an internal error and is unable to
continue normally. Report the following information to technical support.
SOLID Fatal error: Out of central memory when allocating buffer memory (size = 33554432)
Date: 2012-04-24 15:39:44
Product: IBM solidDB
Version: 7.0.0.2 Build 2012-04-20
```

```
[solid1]~ ./solidsma -f -c .
Server could not allocate shared memory segment by id -1
```

Causes

The SMA server startup fails because there is no memory available. This situation can occur if:

- When a SMA application or SMA server terminates abnormally, they can leave shared memory allocated. Even if you shut down all SMA processes, the shared memory is still left reserved.
- You have allocated too little memory for SMA use.

This leads to a situation where all memory is used and you cannot start a SMA server any more.

Resolving the problem

In Linux and UNIX environments, clear the hanging shared memory segments with the `ipcrm` command.

For example in Linux environments, use the following script to identify and remove the unused shared memory segments.

```
#!/bin/sh

if [ $# -ne 1 ]
then
   echo "$0 user"
   exit 1
fi

for shm_id in $(ipcs -m|grep $1|awk -v owner=$1 ' { if ( owner == $3 ) {print $2} }')
do
  ipcrm -m $shm_id
done
```

For more details on the `ipcrm` command, see your operating system documentation.

Cannot map shared memory area

Symptoms

When trying to connect to a SMA server, the following type of error is displayed, and the connection fails.

- Linux and UNIX operating systems

  ```
  cannot map shared memory area 1288077395 to 0x2b0029800000
  Cannot connect to target database.
  ```

- Windows operating systems

  ```
  SQL State "08004"; Native Error Code "25215";
  Error Text "SMA failed in MapViewOfFileExt,
  desired addr: 0000000800000000, got addr: 000000000000000000, error: 6.
  ```

Causes

When started, the SMA starts attaching shared memory segments to an address space that is used by another process.

Resolving the problem

In general, the earlier your application connects to the SMA server, the less likely it is that the address space requested by solidDB is in use.

The SMA server uses the following address spaces by default:

Table 47. SMA default address spaces

Operating system	Default start address space*
AIX	0x700000010000000ul
Linux 64-bit	0x2c0000000000
Linux 32-bit	0x50000000
Solaris Intel	0x2b0000000000
Solaris Sparc	0xffffffff60000000
Windows	0x0000000080000000
*The start address space is the value of the parameter **shmaddr** in the shmat() system call.	

1. Force the start address space for the SMA server to a different address space using the environment variable SOLSMASTART.
   - Linux and UNIX operating systems:

     ```
     export SOLSMASTART=<start_address_space>
     ```

     For example:

     ```
     export SOLSMASTART=0x2b0000000000
     ```

   - Windows operating systems:

     ```
     set SOLSMASTART=<start_address_space>
     ```

     For example:

     ```
     set SOLSMASTART=0x0000000800000000
     ```

2. Restart the SMA server.

Error 21300: Protocol 'sma' is not supported

Symptoms

When trying to connect to a SMA server, the following type of error is displayed:

```
Error HY000: SOLID Communication Error 21300:
Protocol 'sma' is not supported
SQLConnect failed
```

Causes

The application has been linked both to the solidDB ODBC library and the SMA library (`ssolidsma`*xx*).

Resolving the problem

Check your application code and remove any references to the solidDB ODBC libraries (for example, `sacl2x70.so` or `socw6470.dll`.

9.1.5 Troubleshooting database file size (file write fails)

If your database has reached the maximum size specified by the **IndexFile.FileSpec** parameter, you need to increase the maximum file size limit or divide the database into multiple files.

Symptom

solidDB goes down with `Error 11003 File write failed, configuration exceeded (SU_ERR_FILE_WRITE_CFG_EXCEEDED)`.

Resolving the problem

1. Add a new database file by using the following command:

   ```
   ADMIN COMMAND 'filespec -a "file_name max_file_size_in_bytes [device_number]"]'
   ```

For example:

```
ADMIN COMMAND 'filespec -a "solid2.db 2147483647"'
```

Note:

- You can add new database files only with the `ADMIN COMMAND 'filespec -a'` command, you cannot modify the size of existing database files.
- The new database file specification is stored in the `solid.ini` configuration file at next shutdown.

or

1. Shut down solidDB.
2. Modify the **IndexFile.FileSpec** parameter in the `solid.ini` file.
   - Increase the maximum limit for the database file.

     or
   - Divide the database into multiple files by using the **FileSpec_[1..n]** format.

     For example:

     ```
     [IndexFile]
     FileSpec_1 = solid.db 2147483647
     FileSpec_2 = solid.db2 2147483647
     FileSpec_3 = solid.db3 2147483647
     ```

     Important: If you have not defined the **FileSpec_1** parameter earlier, use the default file size (2147483647) as shown above.
3. Restart solidDB.

Related information:

"FileSpec_[1...n] parameter" on page 48
The **Indexfile.FileSpec** parameter describes the location and the maximum size of an index file (database file).

F.1, "ADMIN COMMAND," on page 359

9.1.6 Troubleshooting MME.ImdbMemoryLimit

If you get an error message indicating that the limit set with **MME.ImdbMemoryLimit** has been reached, you need to take action immediately.

You must address both the immediate problems and the long term problems. The immediate problems are to prevent users from experiencing serious errors, and to free up some memory before shutting down the server so that your system is not out of memory when you restart the server. For long term, you need to ensure that you will not run out of memory in the future as tables expand.

Resolving the immediate problem

To address the immediate problem, you typically need do the following:

1. Notify users that they should disconnect from the server. This will accomplish two things: it will minimize the number of users who will be impacted if the situation deteriorates. Also, if any of the users who disconnect were using temporary tables, disconnecting will free up memory. You may wish to have a policy or error-checking code to ensure that users and/or programs will attempt to disconnect gracefully if they see this error.
2. If there were not enough temporary tables to free memory, drop some transient table indexes or transient tables if any exist.

If there were not enough temporary tables and transient tables to free enough memory, do the following:

1. Drop one or more indexes on in-memory tables.
2. Shut down the server.
3. If there was absolutely nothing in memory that you could discard (for example, you had only normal in-memory tables, none of which had indexes, and all of which had valuable data), increase the **MME.ImdbMemoryLimit** slightly before restarting the server. This may force the server to start paging virtual memory which will greatly reduce performance, but it will allow you to continue using the server and address the long-term problems. If you previously set the **ImdbMemoryLimit** a little bit lower than the maximum, you will be able to raise it slightly now without forcing the system to start paging virtual memory.
4. Restart the server.
5. Minimize the number of people using the system until you have had time to address the long-term problem. Ensure that users do not create temporary tables or transient tables until the long-term problem has been addressed.

Resolving the long term problem

After you have solved the immediate problem and have ensured that the server has at least some free memory, you are ready to address the long term problems.

For long term, reduce the amount of data stored in in-memory tables. The ways to do this are to reduce the number or size of in-memory tables (including temporary tables and transient tables), or reduce the number of indexes on in-memory tables.

- If the problem was caused solely by heavy usage of temporary or transient tables, ensure that not too many sessions create too many large temporary or transient tables at the same time.
- If the problem was caused by using too much memory for normal in-memory tables, and if you cannot increase the amount of memory available to the server, move one or more tables out of main memory and onto the disk.

To move a table from memory to disk, do the following:

1. Create an empty disk-based table with the same structure (but a different name) as one of the tables in memory.
2. Copy the information from the in-memory table to an intermediate disk-based table.

   If you try to copy records of a large table to another table using a single SQL statement (INSERT INTO ...VALUES SELECT FROM), keep in mind that the entire operation occurs in one transaction. Such an operation is efficient only if the entire amount of data fits in the cache memory of the server. If transaction size outgrows the cache size, the performance degrades significantly. Therefore, you should copy data of a large table to another table in smaller transactions (for example, few thousands of rows per transaction) using a simple stored procedure or application.

   Note: The intermediate table does not need indices. The indices should be re-created in the new table after the data has been successfully copied.
3. Drop the in-memory table.
4. Rename the disk-based table to have the original name of the dropped in-memory table.

Tip:

- You should set the **MME.ImdbMemoryLimit** to a slightly lower value than the maximum you really have available. If you run out of memory and have no unnecessary in-memory tables or indexes that you can get rid of, you can increase the **MME.ImdbMemoryLimit** slightly, restart the server with enough free memory that you can address the long-term need.
- Use the **MME.ImdbMemoryWarningPercentage** to warn you about increasing memory consumption.
- Not all situations require you to reduce the number of in-memory tables. In some cases, the most practical solution may be to simply install more memory in the computer.

9.1.7 Troubleshooting solidDB Data Dictionary (soldd)

soldd returns error 23007 when exporting database schema

Symptom

When exporting database schema, **soldd** returns error 23007.

For example:

```
Solid Data Dictionary List fatal error: [Solid][SOLID ODBC Driver]
[SOLID]SOLID Procedure Error 23007: Procedure name
SOLDD_GET_SEQUENCE_VAL conflicts with an existing entity
```

Causes

The error 23007 is a generic solidDB procedure that is returned when you attempt to create a stored procedure with a name that exists in the database. **soldd** creates system stored procedures to export the current value of a database sequence object and drops the same once the sequence object is exported. If **soldd** is interrupted during the schema export, dropping the system stored procedures might fail. When **soldd** is rerun, error 23007 is returned.

Resolving the problem

1. Check the error message for the name of the system stored procedure that is causing the error.
2. Drop the procedure with the following command:

   ```
   DROP PROCEDURE <procedure name>
   ```
3. Re-export the schema with **soldd**.

Related information:

7.5, “solidDB Data Dictionary (soldd),” on page 155

9.1.8 Troubleshooting encryption and authentication

External authentication requires the use of IBM Global Security Kit (GSKit). If the use of GSKit is not enabled or solidDB server or client cannot load the GSKit library, the server startup or client connection fails.

9.2 Searching knowledge bases

You can find useful information by searching the Information Center, but sometimes you need to look beyond the Information Center to answer your questions or resolve problems.

About this task

To search knowledge bases for information that you need, use one or more of the following approaches:

Procedure

- Find the content that you need by using the IBM Support Portal.

  The IBM Support Portal is a unified, centralized view of all technical support tools and information for all IBM systems, software, and services. The IBM Support Portal lets you access the IBM electronic support portfolio from one place. You can tailor the pages to focus on the information and resources that you need for problem prevention and faster problem resolution.

  The following link provides a list of all solidDB product family TechNotes, ordered by publication date.

  – solidDB product family TechNotes
- Search for content about solidDB products in developerWorks®

  developerWorks is an IBM resource for developers and IT professionals.
- Search for content by using the IBM masthead search. You can use the IBM masthead search by typing your search string into the Search field at the top of any ibm.com® page.
- Search for content by using any external search engine, such as Google, Yahoo, or Bing. If you use an external search engine, your results are more likely to include information that is outside the ibm.com domain. However, sometimes you can find useful problem-solving information about IBM products in newsgroups, forums, and blogs that are not on ibm.com.

  Tip: Include "IBM" and the name of the product in your search if you are looking for information about an IBM product.

9.3 Getting fixes

A product fix might be available to resolve your problem.

About this task

All solidDB fix packs and interim fixes are available through Fix Central (http://www.ibm.com/support/fixcentral/).

Procedure

1. Visit the following solidDB Support page for a list of available fix packs and download links to the installation images: Fix packs by version for solidDB and solidDB Universal Cache
2. Determine which fix pack you need. In general, to avoid encountering problems caused by software defects already known and corrected, the installation of the most recent fix pack is recommended.
3. Download the fix pack and extract the files into a directory of your choice.
4. Apply the fix. Follow the instructions in the `readme.txt` file provided with the fix.

   Tip: You can view and download the `readme.txt` file separately using the Fix Central HTTP download option.

9.4 IBM Software Support for solidDB

For assistance with solidDB product defects, collect relevant diagnostics data and contact IBM Software Support. Before contacting IBM Software Support, your company must have an active IBM software maintenance contract.

9.4.1 Contacting IBM Support

IBM Software Support provides assistance with product defects.

Before you begin

Before contacting IBM Software Support, your company must have an active IBM software maintenance contract, and you must be authorized to submit problems to IBM. For information about the types of available support, see the Support portfolio topic in the *Software Support Handbook*.

Procedure

1. Define the problem, gather background information, and determine the severity of the problem. For more information, see the Getting IBM support topic in the Software Support Handbook.
2. Collect diagnostic information.

   See 9.4.2, "Collecting diagnostics data," on page 205 for details.
3. Submit the problem to IBM Software Support in one of the following ways:
   - Online through the IBM Support Portal: You can open, update, and view all your Service Requests from the Service Request portlet on the Service Request page.
   - By phone: For the phone number to call in your country, see the Directory of worldwide contacts web page.

Results

If the problem that you submit is for a software defect or for missing or inaccurate documentation, IBM Software Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Software Support provides a workaround that you can implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM Support Portal daily, so that other users who experience the same problem can benefit from the same resolution.

Sending information to IBM Support

You can submit data to IBM Software Support by FTP or by using the Electronic Service Request (ESR) tool.

Before you begin

The steps assume that you have already opened a problem management record (PMR) with IBM Software Support.

Procedure

- To submit files (via FTP) to the Enhanced Centralized Client Data Repository (EcuRep):
  1. Package all files into ZIP or TAR format, and name the package according to your Problem Management Record (PMR) identifier.

To associated the file with the correct PMR, use the following naming convention: xxxxx.bbb.ccc.yyy.yyy, where xxxxx is the PMR number, bbb is the PMR branch number, ccc is the PMR territory code, and yyy.yyy is the file name plus the description of the file type tar.Z or xyz.zip.

2. Using an FTP utility, connect to the server ftp.emea.ibm.com.
3. Log in as the user ID anonymousand enter your email address as your password.
4. Go to the toibm directory. For example, cd toibm.
5. Go to one of the operating system-specific subdirectories. For example, the subdirectories include: aix, linux, unix, or windows.
6. Change to binary mode. For example, enter bin at the command prompt.
7. Put your file on the server by using the put command. Use the following file naming convention to name your file and put it on the server. Your PMR will be updated to list where the files are stored. You can send files to the FTP server, but you cannot update them. Any time that you must later change the file, you must create a new file name.
8. Enter the quit command.

- To submit files using the ESR tool:
  1. Sign onto ESR.
  2. On the Welcome page, enter your PMR number in the **Enter a report number** field, and click Go.
  3. Scroll down to the **Attach Relevant File** field.
  4. Click **Browse** to locate the file that you want to submit to IBM Software Support.
  5. Click **Submit**. Your file is transferred to IBM Software Support through FTP, and it is associated with your PMR.

9.4.2 Collecting diagnostics data

Depending on your environment and setup, you can use the solidDB Support Assistant and InfoSphere CDC Support Assistant for collecting diagnostics data. In some cases, you might need to collect the data manually.

solidDB Support Assistant

The Support Assistant (**solidsupport**) utility helps you collect diagnostic files and system information for troubleshooting purposes.

The **solidsupport** utility collects diagnostic files such as solmsg, soltrace, and ssdebug from the database instance in question and stores them in a compressed archive file (solidsupport.zip). The utility also produces directory listings of database, logging, and sorter directories, and collects various operating system and environment-specific information.

Execute the command **solidsupport -h** to display the complete list of command options.

The **solidsupport** utility collects the following information by default:

Table 48. **solidsupport**

Content type	Notes®
solidDB configuration file	Default file name is solid.ini. If solid.ini does not exist or the configuration file name is not provided with option **-i**, the factory values are used.

Table 48. **solidsupport** *(continued)*

Content type	Notes®
Message files • solmsg.out • solerror.out	For more information about message and log files, see 5.1, "Viewing error messages and log files," on page 92.
Network and SQL monitor trace files – soltrace.out	For information about how to enable the generation of network trace files, see "Network trace facility" on page 190.
High Availability Controller (HAC) files • hactrace.out • hacmsg.out • solidhac.ini	The location of the HAC related files must be specified with the option -c *HAC_directory_path*
Debug files • ssdebug.out • ssdebug.log • Stack trace files – ssstacktrace-xxx-yyy.out	The debug files are generated only in exceptional cases. IBM Software Support provides instructions if the debug files are needed. For information about the stack traces facility, see "Using stack trace facility" on page 189.
Performance counter reports • pmondiff.out	The performance counter reports are collected if you have generated such reports with ADMIN COMMAND 'perfmon diff'.
Report files (rep*)	The report files are collected if you have generated such files with ADMIN COMMAND 'report *filename*'. Only file names starting with rep are collected. **Tip:** You can also turn on automatic report file generation with the **Srv.ReportInterval**, **Srv.MemorySizeReportInterval**, and **Srv.DatabaseSizeReportInterval** parameters.
Directory listings of database, logging, backup, and sorter directories	This information is collected into *.list files in the SOLSUPPORT directory in the solidsupport.zip archive.
Operating system and environment information • Operating system patch level • Number of processors • Amount of memory • Swap and file cache settings • User data and file resource limits and per user process limit • Type of disk storage	This information is collected by default into a detailed_system_info.html file. You can also use the option **-f** to specify that instead of HTML output, the collected system information is written into flat text files that are archived into solidsupport_sysinfo.zip file within the main solidsupport.zip archive.

Important: To protect the security of your data, **solidsupport** does not capture any user data from tables or logs by default. To include database and log files and all files from database working directory, use the option **-a**.

Note:

- The **solidsupport** utility collects only existing files; it does not generate any diagnostics files, such as the trace files (soltrace.out). You need to first enable the generations of log files, as described in the **Notes** column in the above table.

- The **solidsupport** utility does not collect any information from the client side (ODBC/JDBC drivers). You need to collect the client-specific information manually; for more information, see section "Collecting client and other diagnostics data" on page 208.

Using solidDB Support Assistant (solidsupport)

Start the Support Assistant (**solidsupport**) with the command solidsupport, followed by argument options.

```
solidsupport [options]
```

*Table 49. solidDB Support Assistant (**solidsupport**) options*

Option	Description
-a	Collects all files from database, log, and working directories, including database files and log files
-c *HAC_directory_path*	Specifies the directory for HAC-related files, default is the solid.ini directory
-o *output_file*	Specifies the output file name Default is solidsupport.zip.
-i *configuration_file*	Specifies the configuration file name and path to be used The configuration file path is used as a working directory for **solidsupport**; all output files are written to this directory. If this option is not given, the default file name solid.ini is used.
-f	Collects system information as flat files and archives them into solidsupport_sysinfo.zip
-m	Collects system information into an HTML file (detailed_system_info.html) – default
-p	Run without pausing
-h	Usage/Help information

The **solidsupport** utility collects data from the machine where it is run. The configuration file path is used as a working directory for **solidsupport**; all output files are written to that directory.

- In a client-server environment, database-related information are collected from the machine where the database resides and from the location specified by solid.ini configuration file.
- In HotStandby setups, you need to run **solidsupport** on both HotStandby nodes.

Example 1

The following command

- checks solidDB file names and paths from the default configuration file solid.ini in the current directory, or, if solid.ini does not exist, the factory defaults are used,
- copies all files to a compressed file with the default name solidsupport.zip

```
solidsupport -a
```

Example 2

The following command

- checks solidDB file names and paths from a configuration file named `solidDB.ini` in the current directory
- copies default set of files to a compressed file named `12345.678.901.zip`

```
solidsupport -o 12345.678.901.zip -i solidDB.ini
```

InfoSphere CDC Support Assistant

The InfoSphere CDC Support Assistant allows you to collect diagnostic data such as configuration, log, and runtime information for Management Console, Access Server, and optionally for specific datastores in your environment. You can also enable trace options for Management Console and Access Server.

For instructions on how to use the InfoSphere CDC Support Assistant, see section **Support and Troubleshooting** > **Using Support Assistant** in the *InfoSphere Change Data Capture Management Console, Administration Guide*.

Collecting client and other diagnostics data

In some cases, IBM Software Support might ask you to collect diagnostics and problem reporting data manually, for example, about your ODBC or JDBC setup.

Gathering diagnostics data on ODBC API: If the problem concerns the performance of a specific ODBC API or SQL statement, run the SQL Info facility at level 4.

The generated `soltrace.out` file contains the following information:

- CREATE TABLE statements
- CREATE VIEW statements
- CREATE INDEX statements
- SQL statements

Related concepts:

"Tracing SQL statements" on page 184
You can trace SQL statements using the ADMIN COMMAND 'trace' and ADMIN COMMAND 'monitor' commands or by using the SQL Info facility.

Gathering diagnostics data on solidDB ODBC Driver: If the problem concerns the performance of the solidDB ODBC Driver, collect the following information:

- solidDB ODBC Driver name and version
- ODBC Driver Manager name and version

If the problem concerns the cooperation of solidDB and any independent software vendor (ISV) software package, include the following information:

- Full name of the software
- Version and language
- Manufacturer
- Error messages from the ISV software package

In Windows environments, you may also use the ODBC trace facility **Administrative Tools** > **ODBC (Data Sources)** > **Tracing** to get a log of the ODBC statements.

Checking solidDB ODBC Driver version

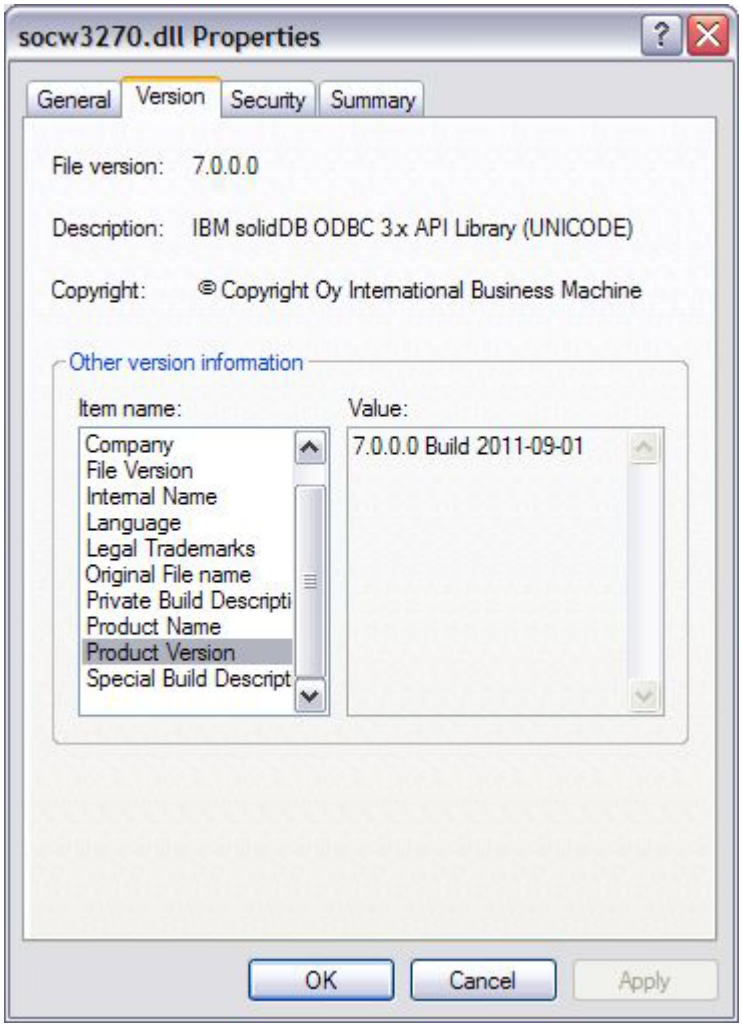
- In Linux and UNIX environments, grep the ODBC driver library file for the string "ODBC 3.x".

  For example:

  ```
  [test1]~% strings /solid/bin/socl2x65.so | grep "ODBC 3.x"
  @(#)IBM solidDB ODBC 3.x API Library (UNICODE) v.6.5.0.4 Build 2011-01-21
  IBM solidDB ODBC 3.x API Library (UNICODE)
  ```

- In Windows environments:
  - Right-click the ODBC driver library file you are using and select **Properties**.

    By default, the ODBC driver library files are in the `bin` directory in your solidDB installation directory.
  - On the **Version** tab, select **Product version**.



Gathering diagnostics data on solidDB JDBC Driver: If the problem is related to the solidDB JDBC Driver, include the following information in your problem report:

- Exact version of JDK or JRE used
- Version of the solidDB JDBC Driver (`SolidDriver2.0.jar`)
- Contents of DriverManager.setLogStream(someOutputStream) output, if available

- Call stack – Exception.printStackTract() output of the application, if an exception has occurred in the application

Checking solidDB JDBC Driver version

1. Navigate to the directory where the solidDB JDBC Driver (`SolidDriver2.0.jar`) is installed.
2. Use the following command to query the solidDB JDBC Driver version:

```
java -cp SolidDriver2.0.jar solid.jdbc.SolidDriver -version
```

The output shows the version information in the following format (example):

```
IBM solidDB JDBC driver 7.0.0.3 Build 2012-11-06
```

Fast path: If you only have one installation of solidDB JDBC Driver, you can also issue the following command:

```
java solid.jdbc.SolidDriver -version
```

Collecting diagnostics data about communication problems 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.

Include also the following information:

- solidDB communication DLLs used: version and size
- Other communication DLLs used: version and size
- Description of the network configuration

9.4.3 Subscribing to Support and other updates

To stay informed of important information about the IBM products that you use, you can subscribe to Support and other updates.

About this task

By subscribing to receive updates, you can receive important technical information and updates for specific Support tools and resources. You can subscribe to updates in the following ways:

- RSS feeds and social media subscriptions

  The following RSS feeds and social media subscriptions are available for solidDB and solidDB Universal Cache:

  – solidDB Support RSS

  – solidDB Product Family forum RSS

- My Notifications

  With My Notifications, you can subscribe to Support updates for any IBM product. You can specify that you want to receive daily or weekly email announcements. You can specify what type of information you want to receive (such as APARs, publications, hints and tips, product flashes (also known as alerts), downloads, and drivers). My Notifications enables you to customize and categorize the products about which you want to be informed and the delivery methods that best suit your needs.

- APARs

  Each APAR enables you to subscribe to receive periodic emails that alert you to the status of the APAR, along with a link to the fix after it becomes available. You can track APARs individually or by product.

Procedure

- To subscribe to RSS feeds, copy the RSS feed URL to your RSS reader.
  - solidDB Support RSS - http://www.ibm.com/software/support/rss/db2/3457.xml?rss=s3457&ca=rssdb2
  - solidDB Product Family forum RSS - http://www.ibm.com/developerworks/forums/rss/rssmessages.jspa?forumID=1310

  For general information about RSS, including steps for getting started and a list of RSS-enabled IBM webpages, visit the IBM Software Support RSS feeds site.
- To subscribe to My Notifications, go to the IBM Support Portal and click **My Notifications** in the Notifications portlet.
- Create or edit your profile to add the solidDB products to your subscriptions list: products to your subscription list.
  - **Software** > **Information Management** > **IBM solidDB**
  - **Software** > **Information Management** > **solidDB product family**

  For more detailed information, see Subscribing to My Notifications support content updates.

Appendix A. Server-side configuration parameters

The server-side configuration parameters define various performance, memory and disk usage, and other characteristics of the solidDB server. Generally, the factory value settings offer the best performance and operability, but in some special cases modifying a parameter might improve performance.

Each section of the `solid.ini` configuration file is documented in a separate table. The sections are:

- Accelerator
- Cluster
- Com
- General
- HotStandby
- IndexFile
- Logging
- LogReader
- MME
- Passthrough
- SharedMemoryAccess
- 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 in-memory databases.
- The Synchronizer section applies only to solidDB advanced replication capability.
- The HotStandby section only applies to the High Availability component.

The descriptions of a some parameters specify that those parameters (or some specific settings of those parameters) apply only to a particular component. Each exception is documented in the description of the parameter itself.

Note: Parameter availability and factory values can vary between platforms.

A.1 Accelerator section

Table 50. Accelerator parameters

[Accelerator]	Description	Factory Value	Access Mode
ImplicitStart	If set to yes, solidDB starts 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 SSC API function SSCStartServer.	yes	RW/ Startup

Table 50. Accelerator parameters (continued)

[Accelerator]	Description	Factory Value	Access Mode
ReturnListenErrors	If this parameter is set to yes and network listening fails, the SSCStartServer function returns an error. If this parameter is set to no and network listening fails, the SSCStartServer starts the LLA server but network connections are not possible.	no	RW/ Startup

A.2 Cluster section

Table 51. Cluster parameters

[Cluster]	Description	Factory Value	Access Mode
ReadMostlyLoadPercentAtPrimary	Defines the percentage of read loads that are directed to the Primary when load balancing is set to READ_MOSTLY.	50	RW/Startup

A.3 Communication section

Table 52. Communication parameters

[Com]	Description	Factory Value	Access Mode
Listen	Defines the network (listening) name for a server. The format of the network name is: `protocol_name [options] server_name` The `options` and `server_name` depend on the communication protocol. For details, see 6, "Managing network connections," on page 117. You can define several network listening names. 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. **Note:** The **`ADMIN COMMAND 'par com.listen=value'`** command does not replace existing network listening names; it appends new listening names to the existing list.	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
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

Table 52. Communication parameters (continued)

[Com]	Description	Factory Value	Access Mode
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
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
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
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 on Linux, HP-UX, and Solaris 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 on Linux, HP-UX, and Solaris 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 52. Communication parameters (continued)

[Com]	Description	Factory Value	Access Mode
TcpKeepAliveProbeCount	This parameter can only be used on Linux, HP-UX, and Solaris 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, and Solaris 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
Trace	If this parameter is set to `yes`, trace information about 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 about 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

A.4 General section

Table 53. General parameters

[General]	Description	Factory Value	Access Mode
BackupBlockSize	Block size for backup file writing **Note:** • The minimum value for **General.BackupBlockSize** is the server block size (defined with **IndexFile.BlockSize** parameter). • The maximum value is 8MB. If the parameter value exceeds the maximum value, the default value is used (64K). • The value of **General.BackupBlockSize** needs to be a multiple of the database block size of the server (defined with **IndexFile.BlockSize** parameter).	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
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
BackupFlushInterval	Specifies the maximum number of blocks (pages) that can be stored in memory before they are flushed to disk during backup operation.	100	RW

Table 53. General parameters (continued)

[General]	Description	Factory Value	Access Mode
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
CheckpointDeleteLog	If this parameter is set to `yes`, 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 (for example, test data created during development). See also the **`BackupDeleteLog`** parameter. **Important:** • If you are using HotStandby and if you set **`CheckpointDeleteLog=yes`** on the Primary server, 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, the Primary will keep the logs even after the data has been checkpointed on the Primary. • If **`LogReader.LogReaderEnabled`** is set to `yes`, the **`CheckpointDeleteLog`** parameter is not effective: the log files are not deleted after a checkpoint. Instead, the log entries are removed only after the log size defined by **`LogReader.MaxLogSize`** has been reached.	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:** **`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.	50000 log writes	RW

Table 53. General parameters (continued)

[General]	Description	Factory Value	Access Mode
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	RW/Startup
DefaultDomainName	On Windows systems, defines the domain name solidDB uses to resolve two-part user IDs of externally authenticated users. Two-part user IDs are composed of a domain and user name (*domain_name**user_name*). If the domain name is specified with the DefaultDomainName parameter, solidDB stores only the *user_name* in SYS_USERS table. Users can also log on to solidDB using *user_name* only. **Note:** This parameter applies to Windows systems only.	no factory value	RO
DefaultStoreIsMemory	If set to `yes`, new tables are created as in-memory tables, unless they are created without an explicit STORE clause in the CREATE TABLE statement. If set to `no`, new tables are stored on disk by default. You can override the factory value by using the STORE clause in the CREATE TABLE statement. **Note:** 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 idle merge.	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	0 on most platforms.	RW/Startup

Table 53. General parameters (continued)

[General]	Description	Factory Value	Access Mode
GSKitLoginRequired	Specifies that to connect to the solidDB server with an ODBC client using a network protocol, the IBM Global Security Kit (GSKit) must be enabled on the (client) computer. When set to `yes`, solidDB server expects that the network connection over which passwords are sent uses strong encryption. This parameter has impact only when internally authenticated users connect to solidDB server over a network connection. The network connection over which the passwords of externally authenticated users are sent always uses strong encryption. Possible values are `yes` (GSKit) and `no`. If **`GSKitLoginRequired`** is set to `yes`, **`Client.UseGSKit`** must be set to `yes` also.	no	RW/Startup

GSKitPath	This parameter defines the path to the directory where the IBM Global Security Kit (GSKit) library is located.
	
	The value of the parameter must be a valid path. For example:
	`[General]`
	`GSKitPath=/home/sol/soliddb-7.0/bin/`
	
	See also **`General.UseGSKit`** and **`General.GSKitLoginRequired`**.
		no factory value	RO

Table 53. General parameters (continued)

[General]	Description	Factory Value	Access Mode
InternalCharEncoding	Starting from version 6.5, this parameter defines the database mode by defining the encoding used for character data types. Possible values are 'raw' and 'UTF8'. • Unicode mode In the Unicode mode, the internal representation for character data types is UTF-8. The internal representation for wide character data types is UTF-16. • partial Unicode mode In the partial Unicode mode, the internal representation for character data types uses no particular encoding; instead, the data is stored in byte strings with the assumption that user applications are aware of this and handle the conversion as necessary. The internal representation for wide character data types is UTF-16. The databases created with solidDB version 6.3 or earlier are of the partial Unicode type. **Important:** The default database mode in 6.5 is partial Unicode. If the value of this parameter is 'raw', the default value of the parameter **Srv.ODBCDefaultCharBinding** is also 'raw'. If the value of this parameter is 'UTF8', the default value of the parameter **Srv.ODBCDefaultCharBinding** is 'locale:'.	Raw	RW/Create
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 **WriterIOThreads** parameter. The **IOThreads** must be greater than **WriterIOThreads**. If this rule is violated, the **IOThreads** parameter takes the precedence (wins).	5	RW/Startup

Table 53. 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 does not affect performance directly 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 (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 needs to be. However, it can be difficult to calculate the number of locks that you need, so you might 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

Table 53. 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 does not start if the default lock timeout in `solid.ini` is more than 1000 seconds.
	Note: You can set the lock timeout for a single connection by using the following SQL command:
	`SET LOCK TIMEOUT timeout_in_seconds`
	
	You can change the granularity of the SET LOCK TIMEOUT command from seconds to milliseconds by appending "MS" to the number. For example:
	`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

LongSequentialSearchLimit	Sets the number of sequential fetches after which search is treated as long sequential search	500	RW/Startup
MaxMergeParts	This parameter is used to specify the maximum number of concurrent merge operations, or the number of merge parts.	5	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
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 available and the more scattered the writes are, the higher the optimal value. The value cannot be higher than the number of available cores (CPUs). Value 0 means that there is no limit on the number of concurrent writes.	0	RW/Startup

Table 53. General parameters (continued)

[General]	Description	Factory Value	Access Mode
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:** **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 8.6.2, "Setting the MergeInterval parameter," on page 175.	0	RW
MultiprocessingLevel	This parameter defines the number of processing units (processors, cores) available in the computer system. Typically, the concurrency of write operations in the database can be improved if the value matches the number of physical processors (cores) in your system. As of V6.5 Fix Pack 4, the factory value is read from the system as the number of logical processing units. The auto-detected value is output to `solmsg.out` at server startup. With some processor architectures, the number of logical processing units might not be the same as the number of physical cores. In such cases, the optimal value for this parameter typically varies between the number of the physical cores and the number of logical processing units. In releases before V6.5 Fix Pack 4, the factory value of this parameter is 4. **Note:** As of V6.5 Fix Pack 4, the value of the **MME.RestoreThreads** parameter defaults to the value of this parameter, unless set to a different value explicitly.	Read from system	RW/Startup
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 60 seconds use value 60000 (milliseconds). 0 (no timeout)	30000	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

Table 53. General parameters (continued)

[General]	Description	Factory Value	Access Mode
NetBackupCopySolmsgOut	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
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 60 seconds use value 60000 (milliseconds).	30000	RW/Startup
NetBackupReceiveBufferSize	Defines the buffer size at the NetBackup server for storing data during backup. When the buffer is full, writes of backup data are throttled.	`32M` Unit: 1 byte k=KB m=MB	RW
PAMServiceName	The **`General.PamServiceName`** parameter defines the solidDB program name that is used in the PAM configuration to define how solidDB users are authenticated. **Note:** This parameter applies to Linux and UNIX systems only.	solid	RO
Pessimistic	If set to `yes`, the server uses pessimistic concurrency control for D-tables. With 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'; D-tables uses optimistic concurrency control by default. When setting the **`Pessimistic`** parameter to 'yes', the server defaults to pessimistic locking for any new tables that are created and for any old tables for which the concurrency control method was never explicitly set with the ALTER TABLE command. If you set the locking mode of a table by using the following ALTER TABLE command, it takes precedence over this parameter. `ALTER TABLE base_table_name SET {OPTIMISTIC | PESSIMISTIC}`	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

Table 53. General parameters (continued)

[General]	Description	Factory Value	Access Mode
Readonly	If set to `yes`, the database is set to read-only mode. **Tip:** You can query the read-only mode of the database by using the **`ADMIN COMMAND 'getreadonlyflag'`** command.	no	RW/Startup
RSAKeySize	Defines the RSA algorithm key length that is used in IBM Global Security Kit (GSKit) encryption. The unit is bits. Possible values are 1024, 2048, and 4096.	1024	RO
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	This parameter 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, the **`TableLockWaitTimeout`** setting provides wait period of the transaction 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

Table 53. General parameters (continued)

[General]	Description	Factory Value	Access Mode
TransactionEarlyValidate	When set to `yes`, early validation of transaction is used (transaction is validated at the time each statement is written, not at commit time). Early validation is applicable with optimistic locking only. The possible values are `yes` and `no`.	yes	RW/Startup
TransactionHashSize	The hash table contains slots that are occupied by incomplete (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. For example, 5 is a good initial limit for the transaction per slot average. **Note:** 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 Minimum value is 1000.	1046527	RW/Startup
UseEncryption	This parameter defines whether passwords are encrypted. If set to 'no', passwords are not encrypted. For more details, see 4.2, “Encryption,” on page 76.	yes	RW/Startup
UseGSKit	This parameter defines whether IBM Global Security Kit (GSKit) is used for encryption of passwords and database and log files. For more details, see 4.2.1, “Enabling encryption with IBM Global Security Kit (GSKit),” on page 76. **Note:** If **General.UseEncryption** parameter is set to `no`, this parameter is not effective. See also **General.GSKitPath**.	no	RO
VersionedPessimisticReadCommitted	If this parameter is enabled, pessimistic D-tables with READ COMMITTED isolation use versioned reads. Reads with SELECT FOR UPDATE work as before. In other words, pessimistic D-tables work like M-tables.	yes	RW/Startup

Table 53. General parameters (continued)

[General]	Description	Factory Value	Access Mode
VersionedPessimisticRepeatableRead	If this parameter is set to `yes`, pessimistic D-tables with REPEATABLE READ isolation use versioned reads.	yes	RW/Startup
WriterIOThreads	Number of helper threads dedicated to writing tasks (per IO device). **Note:** The **`IOThreads`** must be greater than **`WriterIOThreads`**. If this rule is violated, the factory value is used. If **`IOThreads=1`** then the setting **`WriterIOThrreads=0`** is enforced.	1	RW/Startup

A.5 HotStandby section

Table 54. HotStandby parameters

HotStandby	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
`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. 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`, the server is automatically put in PRIMARY ALONE state (rather than PRIMARY UNCERTAIN state) when the connection to the Secondary is broken.	`no`	RW

Table 54. HotStandby parameters (continued)

HotStandby	Description	Factory value	Access mode
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. Valid values are 1-99.	50	RW
Connect	The **Connect** parameter indicates the address of the other HotStandby server in the pair. The value of this parameter is a standard solidDB connect string (Basic Connectivity) or a TC-specific connect string (Transparent Connectivity). The connect string defined with this parameter must match the server listening name of the other HotStandby server (defined with **Com.Listen** parameter). If you omit this parameter in a server that you intend for HotStandby, 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 HotStandby connection, that is, PRIMARY ALONE, SECONDARY ALONE, and STANDALONE. If **HSBEnabled** is set to no, this parameter is ignored. For Transparent Connectivity (TC) connections with multi-home servers, the **Connect** parameter can be overridden with the **HotStandby.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 is used with the following administration commands:
	• **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

Table 54. HotStandby parameters (continued)

HotStandby	Description	Factory value	Access mode

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.
	
	The **ADMIN COMMAND 'hotstandby netcopy'** is the recommended way to copy the database because it is a more flexible solution.
		No factory value	RW

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	Defines the 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
NetcopyReceiveBufferSize	Defines the buffer size at Secondary server for storing data during netcopy. When the buffer is full, writes of netcopy data are throttled.	`32M` Unit: 1 byte k=KB m=MB	RW
NetcopyRpcCompress	Controls whether data compression is used for a **netcopy** connection.	`no`	RW
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

Table 54. HotStandby parameters (continued)

HotStandby	Description	Factory value	Access mode
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. Possible values are: 1safe, 2safe and auto 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.	2-safe	RW
SecondaryThreads	This parameter defines the number of threads that the Secondary server uses for processing write operations. The optimal number of threads depends on the environment. In principle, Valid values are 1–256.	4	RW/Startup
TCConnect	This parameter defines the address of the other HotStandby server in the pair for a Transparent Connectivity (TC) connection, if the applications and servers need to use different networks to connect to each other (for example, when using multi-home servers). From the application connection perspective, the address specified with this parameter precedes the address defined with the **HotStandby.Connect** parameter. The TC connection will thus use the server addresses specified with this parameter, while the HotStandby connection between the servers uses the server addresses defined with the **HotStandby.Connect** parameter.	No factory value.	RW

A.6 IndexFile section

Table 55. 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

Table 55. IndexFile parameters (continued)

[IndexFile]	Description	Factory Value	Access Mode
BonsaitreeJoinLimit	Defines the size in percentages for the temporary storage (BonsaiTree) index pages after which a join is applied. The default size is half of the value of **IndexFile.BtreeJoinLimit**. For example, if **IndexFile.BtreeJoinLimit** is set to 48, the default **IndexFile.BonsaitreeJoinLimit** value is 24. Minimum value is 0, maximum 50.	20	RW/Startup
BtreeJoinLimit	Defines the size in percentages for the permanent storage (B-tree) index pages after which a join is applied. Minimum value is 0, maximum 50.	40	RW/Startup
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 with the ADMIN COMMAND. For example: `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 has only a small amount of free memory available, you reduce the **CacheSize** value.	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. If each block is 8 KB, the value of 500 corresponds to 4 MB of disk space.	500	RW/Startup

Table 55. IndexFile parameters (continued)

[IndexFile]	Description	Factory Value	Access Mode
FileSpec_[1... N]	Defines the location, name, and the maximum size of the index file. In solidDB, the term *index file* is used as a synonym for *database file*. The parameter accepts the following arguments: • database file location - default is solidDB working directory • database file name • maximum size (in bytes) • device number The device number is an optional argument that defines 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. For example: `FileSpec_1=c:\soldb\solid.db 200000000` 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 more information, see "FileSpec_[1...n] parameter" on page 48. 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. You can also 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)	RO
PreFlushPercent	Sets the percentage of page buffer which is kept clean by the preflush thread. 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.	25	RW/Startup
ReadAhead	Sets the number of prefetched index reads during long sequential searches. 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 improves the overall performance of sequential searches.	4	RW/Startup

Table 55. 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 collision, use the **ReferenceCacheSizeForHash** parameter to accommodate the enlarged cache. The **ReferenceCacheSizeForHash** parameter value is used for calculating the cache hash table size. Use the parameter if you know in advance what the maximum cache size is during the server lifecycle. 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	RW/Startup

A.7 Logging section

Table 56. Logging parameters

[Logging]	Description	Factory Value	Access Mode
BlockSize	Sets the block size of log files. The log block size can be changed between startups. Logs that have a block size different than the one set with this parameter are accepted at recovery. The value must be a multiple of 1 KB. Large blocks reduce the overhead of log writing.	16 KB Unit: byte k=KB	RW/Startup
DigitTemplateChar	Specifies the template character that is replaced in the name template of the log files. See the description of the **Logging.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 56. Logging parameters (continued)

[Logging]	Description	Factory Value	Access Mode

DurabilityLevel	This parameter controls whether the transaction durability level is strict, relaxed, or adaptive.
	• 1 = relaxed durability
	If durability is *relaxed*, writes are asynchronous; there can be a delay between the time that the transaction is committed and the time that it is logged.
	• 2 = adaptive durability
	This value applies only to HSB (HotStandby) Primary servers.
	• 3 = strict durability
	If durability is *strict*, writes to the transaction log are synchronous; as soon as a transaction has been committed, the transaction is written to the transaction log.See 8.1, "Logging and transaction durability," on page 163 for more details.
	
	The durability level can be set dynamically by using the command:
	`ADMIN COMMAND`
	`'parameter Logging.DurabilityLevel=n';`
	
	where `n` is one of the valid values for this parameter.
	
	You can override the parameter setting for each connection or session by using the SET DURABILITY or SET TRANSACTION DURABILITY command.
	
	The **`DurabilityLevel`** parameter affects the server behavior only if transaction logging is turned on. If you turn off transaction logging by setting **`Logging.LogEnabled`** to `no`, your data is not be logged to disk, regardless of the setting of **`DurabilityLevel`**. If **`Logging.LogEnabled`** is set to `no` and **`DurabilityLevel`** is set, the server displays a warning message at the time that it starts.
	
	See also **`Logging.LogWriteMode`** and **`HotStandby.2SafeAckPolicy`** parameters.
		1	RW

FileFlush	This parameter controls the log file flush behavior. This parameter is only valid for platforms supporting Synchronized I/O Data Integrity Completion, such as Solaris, HP-UX, and Linux. When set to `no`, the operating system flushes the log file, instead of the solidDB engine.	yes	RW/Startup

Table 56. Logging parameters (continued)

[Logging]	Description	Factory Value	Access Mode
FileNameTemplate	Defines the path and naming convention used when creating log files. The log files contain information that is used to recover data if the server terminates abnormally. At minimum, this parameter defines the naming convention used when creating log files. The path information is optional. If the path is not defined, the **Logging.LogDir** parameter defines the path. Template characters (for example, #) are replaced with sequential numbers. For example, the following file entry instructs the server 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 can use 4 - 10 template characters. If you do not want to use the "#" sign as a template character, you can specify a different character by setting the parameter **Logging.DigitTemplateChar**. After the last digit in the log file name sequence is reached (for example, `sol99999.log`), the server restarts the numbering from 0 (for example, `sol00000.log`). To achieve better performance by avoiding problems with network I/O, store the log files on a local drive using local disk names.	sol#####.log	RW/Startup
LogDir	This parameter sets the directory prefix of the log file path, if it is not specified with the **Logging.FileNameTemplate** parameter. By default, **Logging.FileNameTemplate** specifies only the file name and **Logging.LogDir** specifies the path, which is the servers working directory. The specified log directory has to exist prior to starting the server. If the directory does not exist, the server returns the following type of error: `SsBOpenLocal failed, file 'log/sol00001.log', errno = 2, retries = 0, open files = 1`	"." (the server's working directory)	RW/Startup
LogEnabled	Specifies whether transaction logging is enabled or not. If transaction logging is disabled, you get better performance but lower transaction durability. If logging is disabled and the server shuts down unexpectedly, you lose any transactions since the last checkpoint. This parameter applies to in-memory tables and disk-based tables.	yes	RW/Startup
LogSoftMemoryLimit	Limits the amount of memory write queue items (writeq) can use. The limit is a soft limit: it can be exceeded temporarily, for example, in case of large write operations. The minimum value is 1 MB. **Note:** In V7.0 Fix Pack 6, the default value is 4 M. If you experience performance degradation during heavy write loads after installing V7.0 Fix Pack 6, increase the value of the **Logging.LogSoftMemoryLimit** parameter, for example, to 64 M.	67108864 (64 M) Unit: 1 byte or k=KB m=MB	RW
LogWriteMode	Specifies the mode in which the log is 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, see 2.7.10, "Transaction logging," on page 32.	2 (Overwrite method)	RW/Startup

Table 56. Logging parameters (continued)

[Logging]	Description	Factory Value	Access Mode
MinSplitSize	Defines the log file size after which the log is written to a new log file. The new log file is created after the next checkpoint.	10 MB Unit: 1 KB k=KB m=MB	RW/Startup
RelaxedMaxDelay	This parameter sets the maximum time in milliseconds that the server waits until the committed transactions are written to the log. This parameter applies only when the transaction durability level is set to RELAXED with the **Logging.DurabilityLevel=1** parameter or the SET DURABILITY statement. The units are milliseconds. Minimum allowed value: 100 (100 ms).	5000 milliseconds (5 seconds) Unit: 1 ms	RW/Startup
SyncWrite	When set to yes, the solidDB server assumes that the platform supports Synchronized I/O Data Integrity Completion. This parameter applies only to platforms, such as Solaris, HP-UX, and Linux, which support Synchronized I/O Data Integrity Completion. It must be set to `no` on all other platforms.	no	RW/Startup

ThreadPriority	This parameter defines the logging thread priority in Linux, AIX, Solaris, and HP-UX environments. Defining the logging thread priority can increase throughput in case of write-intensive workloads when solidDB is under high load.
	
	Possible values are:
	• `Normal` (default) - Logging thread is not prioritized.
	• `Realtime` - Logging thread is prioritized using the realtime scheduler of the operating system.
	• `Adaptive` - Server tries to set the logging thread priority as realtime. If the realtime priority cannot be set, the server defaults to `Normal` priority.
	
	If you configure the parameter as `Realtime`, but the server is unable to do so, the server does not start.
	
	If the parameter is set to `Adaptive` and the server fails to set the thread as realtime at startup, an error is output in `solmsg.out`.
	Note: Before setting the **Logging.ThreadPriority** parameter to `Realtime` or `Adaptive`, you might need to adjust the security settings for realtime use in your environment.
	
	For example, in Linux 64-bit environments:
	1. Add the following two lines in `/etc/security/limits.conf`:
	`* hard rtprio 99`
	`* soft rtprio 99`
	2. To activate the new settings, log out and log in again; the limit settings are set per login, they exist only for the duration of the session.
	3. Restart the server.
		Normal	RW

A.8 LogReader section

Table 57. Log Reader parameters

[LogReader]	Description	Factory Value	Access Mode
LogReaderEnabled	By using this parameter, you can enable or disable the log reader capability. In Universal Cache and InfoSphere CDC replication setups, 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 after 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 57. Log Reader 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 the solidDB server: 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, the solidDB server continues the processing and log files are preserved at least until the defined **MaxLogSize** limit is reached (see above).	100000	RW
MaxMemLogSize	Maximum size of the Log Reader logfile in memory, when logging is not enabled (**Logging.LogEnabled = no**). After maximum size is reached, logreader catchup might not be possible anymore. Unit: megabytes.	1 MB	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
UseThrottling	Controls whether the log reader uses throttling to block operations until there is space in the log reader buffer.	yes	RW/Startup

A.9 MME section

Table 58. 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. In-memory tables includes Temporary Tables and Transient Tables, as well as 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`
	
	Value 0 means "no limit".
	
	As a general rule, for servers with 1 GB or less 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.
	Note: This parameter only applies only to solidDB main memory engine tables. It does not apply to disk-based tables.
	
	You can change this parameter 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.
	Important: 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, and so on, and will return error codes.
		0
		
		Unit: 1 byte k=KB m=MB g=GB
			RW

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, and so on. 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). **Note:** This parameter only applies to solidDB main memory engine tables.	90	RW
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. **Note:** This parameter only applies to solidDB main memory engine tables. It does not apply to disk-based tables.	80	RW

Table 58. MME parameters (continued)

[MME]	Description	Factory Value	Access Mode
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 this parameter 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 also the parameter **`LockEscalationLimit`**. Possible values are `yes` and `no`. **Note:** This parameter applies to in-memory tables only.	no	RW/Startup
LockEscalationLimit	If **`LockEscalationEnabled`** is set to `yes`, 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 (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 may 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 58. MME parameters (continued)

[MME]	Description	Factory Value	Access Mode
MaxTransactionSize	This parameter defines the maximum approximate size of a transaction in bytes. Some MME transactions (for example, DELETE FROM <table>) might cause solidDB to allocate a lot of memory for the operation. This can lead to an out-of-memory situation where solidDB cannot allocate any more memory from the operating system, and performs an emergency exit. To prevent this, use this parameter to define the maximum approximate size (in bytes) for each MME transaction; when the transaction size exceeds the value set with this parameter, the transaction fails with the error `SOLID Database error 16509: MME transaction maximum size exceeded`. Value 0 means unlimited.	0	RW

MemoryPoolScope	This parameter sets the memory pool scope. Possible values are `Global` and `Table`.
	
	When set to `Table`, only objects that belong to the same database table are allocated from a single memory segment. This ensures, for example, that dropping a whole table frees the memory segment back to operating system. Only unused memory segments can be returned back to system.
	
	When set to `Global`, memory pools are shared between all MME data.
	
	When **`MME.MemoryPoolScope`** is set to `Table`, you can use the DESCRIBE <table> statement to view the memory consumption for the table. For example:
	`DESCRIBE tmemlimit_tab;`
	`RESULT`
	`------`
	`Catalog: DBA`
	`  Schema: DBA`
	`    Table: TMEMLIMIT_TAB`
	`      Table type: in-memory`
	
	`      Memory usage: 7935 KB (total), 7925 KB (active),`
	`                    6192 KB (rows), 1733 KB (indexes).`
	
	`...`
	`1 rows fetched.`
		Global	RW/Startup

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`, at shutdown, the server releases the memory used by M-tables explicitly, rather than relying on the operating system to clean up all memory associated with this process. Some operating systems 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

Table 58. MME parameters (continued)

[MME]	Description	Factory Value	Access Mode
RestoreThreads	This parameter defines the maximum number of threads used while restoring in-memory database during database startup. If you do not set this parameter explicitly, the value of this parameter is set to the same value as **General.MultiprocessingLevel**. Possible values are between 1 and 65536. Value 1 means that the load is executed in single thread. With invalid values, this parameter defaults to the value of **General.MultiprocessingLevel**. In-memory database restore assigns one thread per each table if the number of tables is smaller or equal to the number of the parameter value. Maximal concurrency is reached when the parameter value is smaller than the following two values: number of cores/processors, and the number of tables in the database.	Same as **General. Multiprocessing Level**	RW/Startup

A.10 Passthrough section

Table 59. SQL passthrough parameters

[Passthrough]	Description	Factory value	Access mode
ComplexNumNonindexedConstr	This parameter specifies the minimum number of non-indexed WHERE clause constraints in a complex statement. If a statement has less non-indexed constraints of the following type, the statement is not complex and it is not passed through to the back end: the WHERE clause constraint does not resolve with index, the index does not exist, or the optimizer chooses different index for constraint. Value 0 (zero) means that number of non-indexed constraints is not used when estimating if the statement is complex. This parameter is effective only when the passthrough mode is CONDITIONAL. Use the performance counter *Passthrough complex by num non indexed constraints* to monitor the number of statements that are passed through when this parameter is set.	0	RW
ComplexNumOrderedRows	This parameter specifies the minimum estimated number of rows which must be sorted in a complex statement. If a statement has less than the estimated number of sortable rows, the statement is not complex and it is not passed through to the back end. Value 0 (zero) means that number of sortable rows is not used when estimating if the statement is complex. This parameter is effective only when the passthrough mode is CONDITIONAL. Use the performance counter *Passthrough complex by num ordered rows* to monitor the number of statements that are passed through when this parameter is set.	0	RW

Table 59. SQL passthrough parameters (continued)

[Passthrough]	Description	Factory value	Access mode
ComplexNumTables	This parameter specifies the minimum number of tables in a complex statement. If a statement has less tables than specified with this parameter, the statement is not complex and it is not passed through to the backend. Value 0 (zero) means that number of tables is not used when estimating if the statement is complex. This parameter is effective only when the passthrough mode is CONDITIONAL. Use the performance counter *Passthrough complex by num tables* to monitor the number of statements that are passed through when this parameter is set.	0	RW

ErrorMapFileName	Specifies the file path and file name for mapping backend native error codes to solidDB error codes.
	`<file_path><file_name>`
	
	For example:
	`[Passthrough]`
	`ErrorMapFileName=myfiles/db2tosoliderrors.txt`
	
	If **`ErrorMapFileName`** is not defined or the error is not mapped, the native backend error codes are mapped to solidDB error 13456 (Passthrough backend error: SQLState=<value>, NativeError=<backend error identifier>, MessageText=<backend error description>).
	
	The entries in the mapping file have the following format:
	`<backend_error> <solidDB error>  ; rest of the line is comment`
	
	As in the `solid.ini` configuration file, semicolon can be used to add comments.
	
	Example:
	`; this file maps DB2 native errors to solidDB native errors`
	`-207 13015 ; column not found`
	`-407 13110 ; NULL not allowed for non NULL column`
	`; end of errormappings`
	
	For more examples on the mapping files, see the `samples/sqlpassthrough` directory in the solidDB installation directory.
		No factory value.	RW/ Startup

Force32bitODBCHandles	The **`Force32bitODBCHandles`** parameter is needed in 64-bit environments when the backend data server is DB2 for Linux, UNIX, and Windows and the IBM Data Server Driver for CLI and ODBC is used with direct linking. When set to `yes`, solidDB server treats the ODBC handles as 32-bit integers instead of the 64-bit void pointers that are native on the 64-bit platforms.	no	RW/ Startup
IgnoreOnDisabled	The **`IgnoreOnDisabled`** parameter defines how the application program perceives the fact that passthrough is disabled. If the value is `yes`, all the statements related to passthrough (`SET PASSTHROUGH ...`) are ignored. If the value is `no`, an error is return on any effort to execute those statements. Possible values are `yes` and `no`.	yes	R/W

Table 59. SQL passthrough parameters (continued)

[Passthrough]	Description	Factory value	Access mode
PassthroughEnabled	The **PassthroughEnabled** parameter defines whether SQL passthrough is enabled or not. • If passthrough is enabled but it cannot be initialized (for example, driver is not found), errors are returned on each effort to pass a statement to the backend. • If the backend server is shut down in a controlled way, the value of the **PassthroughEnabled** parameter can be set dynamically to `no`. The behavior exposed to the applications is then defined with the **IgnoreOnDisabled** parameter. Possible values are `yes` and `no`.	no	RW/ Startup
RemoteServerDriverPath	The **RemoteServerDriverPath** parameter specifies the driver manager path or the driver path for the backend data server specific ODBC driver that solidDB is linked to.		RW/ Startup
RemoteServerDSN	The **RemoteServerDSN** parameter specifies the data source name (if driver manager is used) or the connect string for the backend data server specific ODBC driver that solidDB is linked to. The connect string must in the format of the ODBC call SQLConnect(), as ServerNam.		RW/ Startup
SqlPassthroughRead	The **SqlPassthroughRead** parameter defines how read statements are passed from the solidDB server to the backend. Possible values are 'None', 'Conditional', and 'Force'.	none	R/W
SqlPassthroughWrite	The **SqlPassthroughWrite** parameter defines how write statements are passed from the solidDB server to the backend. Possible values are `none`, `conditional`, and `force`.	none	R/W

A.11 SharedMemoryAccess section

Table 60. Shared memory access parameters

[SharedMemoryAccess]	Description	Factory value	Startup
MaxSharedMemorySize	This parameter sets the maximum total size of the shared memory area used by solidDB. If the SMA server tries to allocate more, an "out of memory" error occurs. With value "0", the maximum value is set automatically to be the size of the physical memory of the computer (platform specific). **Note:** The value set with the **SharedMemoryAccess.MaxSharedMemorySize** parameter takes precedence over the value set with any corresponding kernel parameter (for example, SHMALL in Linux environments). Thus, the value set with the **SharedMemoryAccess.MaxSharedMemorySize** parameter must not be higher than the value set with the corresponding kernel parameter. If you set the **SharedMemoryAccess.MaxSharedMemorySize** parameter, do not use the **Srv.ProcessMemoryLimit** parameter.	0 (automatic) Unit: 1 byte, G=GB, M=MB, K=KB	RW

Table 60. Shared memory access parameters (continued)

[SharedMemoryAccess]	Description	Factory value	Startup
SharedMemoryAccessRights	This parameter sets a validation context for the user access to the shared memory area. The validation context is modeled after a traditional file validation mask. The possible values are: • `user` – access is granted only to the same user as the one that started the SMA server • `group` – access is granted to any user belonging to the same group as the one that started the SMA server • `all` – access is granted to all users	group	RW

A.12 Sorter section

Table 61. 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 be used for sorting. The valid values range from 10% to 50%. For example, if the **`IndexFile.CacheSize`** parameter is set to 20MB, and if **`MaxCacheUsePercent`** is 25, 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	500	RW/Startup
MaxMemPerSort	This parameter sets the maximum memory in bytes that is available for one sort (sorting the result set of one query). The value of this parameter must not exceed the amount of memory available to the sorter - see **`MaxCacheUsePercent`** for more information.	114688	RW/Startup

Table 61. Sorter parameters (continued)

[Sorter]	Description	Factory Value	Access Mode
SorterEnabled	This parameter enables or disables the usage of the external sorter. The external sorter algorithm is used for sorting processes that do not fit in main memory.	Yes	RW/Startup

TmpDir_[1... N]	This 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`
	
	Note: When this parameter is specified in the configuration file, the external sorter algorithm is enabled.
		Defaults to ".",
		
		(The current directory, that is, the directory from which the server was started.)
			RW/Startup

A.13 SQL section

Table 62. SQL parameters

[SQL]	Description	Factory Value	Access Mode
AllowDuplicateIndex	If set to `yes`, allows duplicate index definitions. This parameter provides compatibility with earlier versions. In versions preceding 4.5, it was possible to create duplicate indexes.	no	RO
AuditTrailEnabled	If set to `yes`, audit trail is enabled. Possible values are `yes` and `no`.	no	RO
CharPadding	When set to `yes`, solidDB 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). **Note:** • This parameter is effective only when using ODBC or JDBC drivers, not when using solidDB SQL Editor (**`solsql`**). • This parameter affects the ODBC and JDBC driver behavior. • This parameter is not effective in Unicode databases (**`General.InternalCharEncoding=UTF8`**).	no	RO

Table 62. 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:** This parameter does not force the optimizer to convert OR operations to UNION operations; it only sets a maximum limit on the number of OR operations that the server may convert to UNION operations.	10	RO
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 set to `no`, hints are disabled. For details on hints, see *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 to ODBC v.3.	no	RO

Table 62. 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
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`. Because the `soltrace.out` file can contain information from several sources, set the **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.	100 MB	RW/Startup
IsolationLevel	Possible values: 3 (SERIALIZABLE) 2 (REPEATABLE READ) 1 (READ COMMITTED) For more information about transaction isolation levels, see *SET TRANSACTION ISOLATION* in the *IBM solidDB SQL Guide* and section 8.2, “Choosing transaction isolation levels,” on page 165. **Important:** In-memory tables support only the READ COMMITTED and REPEATABLE READ isolation levels.	1 (Read Committed)	RW
Latin1CaseSemantics	If set to `no`, uppercase/lowercase conversions are disabled for characters with decimal value between 126 and 256.	yes	RW/Startup

Table 62. 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 — for example, "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 VARCHAR, 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. For example, 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
PreferExactNumericFunctions	Controls the precision of SUM() and AVG() type functions. When set to `yes`: • If the argument of SUM() or AVG() function is of exact numeric datatype (TINYINT, SMALLINT, INTEGER, BIGINT, NUMERIC or DECIMAL), the function returns the result in DECIMAL data type. The default precision of DECIMAL is 52 and scale is floating. • Functions FLOOR() CEILING() and ABS() return their result in the same data type as the argument. **Note:** The **SQL.PreferExactNumericFunctions** parameter can be set only by editing the `solid.ini` file.	no	RW/Startup

Table 62. SQL parameters (continued)

[SQL]	Description	Factory Value	Access Mode
ProcedureCache	Specifies the number of procedures which set the size of cache memory for parsed procedures.	10	RW/Startup
SimpleOptimizerRules	When set to `yes`, simple optimization rules are used instead of using full optimization rules.	no	RO
SortArraySize	This parameter sets the size of the array that SQL uses when ordering the result set of a query. The units are "rows" — for example, if you specify a value of 1000, the server will create an array big enough to sort 1000 rows of data.	4000	RW
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
TimestampDisplaySize19	If set to `yes`, the precision (maximum number of digits) of data type timestamp is set to 19. In this case, the timestamp is presented as *yyyy-mm-dd hh:mm:ss*.	no	RO
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 double quotation marks 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

A.14 Srv section

Table 63. 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

Table 63. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
AllowConnect	If set to no, only connections from solidDB Remote Control (**solcon**) or solidDB SQL Editor (**solsql**) are allowed	yes	RW/Startup

At	This parameter can be used specify commands for automating administrative tasks, such as executing operating system commands, creating backups, checkpoints, and database status reports.
	
	The syntax of the value for this parameter is the following:
	`At = At_string`
	`At_string ::= timed_command`
	`[ ,timed_command ]`
	
	`timed_command ::=`
	`[ day ] HH:MM argument`
	`day ::= sun |`
	`mon |`
	`tue |`
	`wed |`
	`thu |`
	`fri |`
	`sat`
	
	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 **General.BackupDirectory** parameter.
	
	If the day is not given, the command is executed daily.
	
	There is no factory value for this parameter.
	
	For more information about entering time commands, including a list of the available commands and their arguments, see section 2.9, "Entering timed commands," on page 34.
		(no factory value)	RW

ConnectionCheckInterval	This parameter specifies the number of seconds between connection status checks in thread/client mode. To use this parameter, you must set the **Srv.ReadThreadMode** parameter to 0 and the **Srv.Threads** to a large enough value to accommodate the threads in your environment. When the **Srv.ReadThreadMode** parameter is set to 2 (default), the server does not detect a broken connection until it tries to write something back to the client.	10 Unit: seconds	RW/Startup
ConnectTimeOut	Specifies the continuous idle time in minutes after a connection is dropped; negative or zero value means infinite. **Note:** The value set with this parameter is not effective for the SMA handshake connection that is used to pass the shared-memory segment handle to the SMA driver.	480 Unit: 1 min	RW/Startup

Table 63. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
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
DisableOutput	Disables generation of the `solmsg.out` and the `solerror.out` files. For details on these files, read 5.1, "Viewing error messages and log files," on page 92. 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 it is not included in the `solid.ini` file, the log files are generated.	no	RO
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 (prefetched) 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 prefetching 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`** parameter.	2	RW/Startup

Table 63. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
ForceThreadsToSystemScope	This parameter applies only to symmetric multiprocess (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 Solaris environments, *threads* are lightweight processes. The value `yes` can improve the server's performance in a multi-CPU machine significantly. The actual performance improvement depends on how evenly the workload is already spread across your CPUs. If this parameter is set to `no`, you should get slightly better performance in single-CPU systems. When this parameter is set to `yes`, it forces lwp threads to be run in system scope, instead of process scope. It also allows Solaris to schedule solidDB threads on any available CPU. This reduces bottlenecks and enhances the parallelization of operations, including I/O. For more information on lwp, see Solaris operating system documentation.	Solaris: yes Other environments: no	RW/Startup
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

Table 63. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
InifileLineSplitting	Defines whether in `solid.ini` configuration file lines that are longer than 79 characters are split into multiple lines when server saves the file. For example, if you create comments longer than 79 characters, the server splits the comments on separate lines using a backslash (\) at the end of the line but without adding a comment marker (;) on the new line. The server handles the lines that have been split in this way; however, applications such as watchdogs might see the file as corrupted and thus fail. Value `no` means lines are never split.	yes	RW/Startup
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	RO
LocalStartTasks	Number of server's internal tasks that execute the local background statements that were started with command START AFTER COMMIT (without FOR EACH REPLICA). Valid values range from 1 - 100. **Note:** In this context *task* refers to solidDB's internal tasks, not *thread* or *task* as used in some Real-Time Operating Systems. A task is an operation that has to be executed, such as checkpoint, backup, or SQL statement. In this case, you 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.	2	RW/Startup

Table 63. 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.
	Note: **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 can happen in systems where low-priority connections are not checked often enough to notice that they have been disconnected. However, because the parameter only affects server administrative tasks, not user tasks, the effect is generally small.
		2 (seconds)	RW/Startup

Table 63. 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 you 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
MaxUsers	Defines the maximum number of connections to solidDB. When the number of maximum users has been exceeded, error 14507 is issued and you can connect to solidDB only with solidDB Remote Control (**solcon**). Value 0 means that the maximum number of connections is not restricted.	0	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 63. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
MemoryReportLimit	This parameter defines the minimum size for memory allocations after which reporting to `solmsg.out` is done.	0 (no reporting)	RW/Startup
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

Table 63. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
ODBCDefaultCharBinding	Defines the binding method for character data types. The options are: • `raw` — no data conversion takes place between solidDB server and the client The value raw can be used when you want your database to use the binding used in the 6.3 or earlier versions of solidDB. • `locale` — the current client locale setting is used, also if set by the client system • `locale:` — the current client setting are overridden with a default locale set of the client system The driver calls setlocale() with an empty string which effectively searches for the locale setting set in the system. For example, in Linux environments, the environmental variable LC_CTYPE is checked first and if that is not defined, the environmental variable LANG is searched. • `locale:<locale name>` — the current client systems setting are overridden and the given locale is used The convention for `<locale name>` depends on the operating system. For example, in Linux environments, the locale name for the code page GB18030 in Chinese/China is `zh_CN.gb18030`. In Windows environments, the locale name for Latin1 code page in Finnish/Finland is `fin_fin.1252`. • `UTF8` — UTF-8 binding is enforced regardless of the locale set in the client-side system The factory value depends on the value of the parameter **`General.InternalCharEncoding`**: • If **`General.InternalCharEncoding`** is 'raw', **`ODBCDefaultCharBinding`** is also 'raw'. • If **`General.InternalCharEncoding`** is 'UTF8', **`ODBCDefaultCharBinding`** is 'locale:'.	raw See description	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
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

Table 63. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
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 **ProcessMemoryLimit** and **ProcessMemoryCheckInterval** parameters are interlinked; if the **ProcessMemoryCheckInterval** parameter is set to 0, the **ProcessMemoryLimit** parameter is not effective, that is, there is no process memory limit. See also parameters **ProcessMemoryLowPercentage** and **ProcessMemoryWarningPercentage**.	0	RW
ProcessMemoryHysteresisPercentage	As the amount on memory used crosses different boundaries specified with, for example, the **ImdbMemoryLowPercentage** 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 **ProcessMemoryHysteresisPercentage** parameter is used to set the difference as a percentage value.	5	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** and **ProcessMemoryCheckInterval** parameters are interlinked; if the **ProcessMemoryCheckInterval** parameter is set to 0, the **ProcessMemoryLimit** parameter is not effective, that is, there is no process memory limit. **Note:** You should not set the **Srv.ProcessMemoryLimit** and **Srv.ProcessMemoryCheckInterval** parameters when using SMA. If you need to limit the memory the SMA uses, use the **SharedMemoryAccess.MaxSharedMemorySize** parameter.	1G Unit: 1 byte, G=GB, M=MB, K=KB	RW

Table 63. 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
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 **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 the following type of error:`30146 Failed to create thread 'dnet_clientthread'`	2	RW/Startup

Table 63. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
RemoteStartTasks	Number of Replica server's internal tasks 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. **Note:** In this context *task* refers to the internal tasks of the solidDB server, not *thread* or *task* as used in some Real-Time Operating Systems. A task is an operation that has to be executed, such as checkpoint, backup, or SQL statement.	2	RW/Startup
ReportInterval	Enables automated generation of reports at the given interval (in seconds). Automated reports are named `reptimestamp>.dbg` and output to the solidDB working directory. Value 0 means that reports are not generated automatically.	0	RW
RowsPerMessage	Specifies the number of rows returned from the server in one network message when an SQLFetch call is executed (and there are no prefetched 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	This parameter defines the starting digit of the message log file (`solmsg.out`) name postfix, if the **`KeepAllOutFiles`** parameter is set to `yes`.
	
	For example, if the value is set to 5, the `solmsg.out` files are named as follows:
	`solmsg5.out`
	`solmsg6.out`
	`solmsg7.out`
	`...`
	
	Valid values range from 0 to 999999.
		0	RW/Startup

Table 63. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
StackTraceEnabled	The **StackTraceEnabled** parameter controls the stack trace functionality upon an assertion failure or a signal caused by server malfunction. When set to yes, the stack trace information is output to ssstacktrace-<process_id>-<thread_id>.out file. The following signals invoke the stack traces output automatically: • SIGSEGV • SIGILL • SIGBUS • SIGTRAP • SIGSYS • SIGEMT The stack traces information is produced only about the thread that received the signal. Additionally, you can generate the stack traces information for all currently running threads by sending the server the SIGUSR1 signal. **Note:** The stack trace feature is not available on Windows systems.	no (Linux 64-bit) yes (Linux 32-bit and UNIX systems)	RW/Startup
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 When set to no, the message log files (solmsg.out) use a date format such as 15.10 09:29:40. The solerror.out file uses a date format such as Mon Oct 22 15:16:35 2007.	yes	RO
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 **KeepAllOutFiles** parameter is set to yes. Valid values range from 0 to 999999.	0	RW/Startup

Table 63. Srv parameters (continued)

[Srv]	Description	Factory Value	Access Mode
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 section F.1, "ADMIN COMMAND," on page 359 and the `-m` command-line option in Appendix C, "solidDB command-line options," on page 271. Monitoring uses the file named `soltrace.out` for output. After reaching the maximum size, the following takes place: 1. solidDB deletes any existing file named `soltrace.bak`; 2. solidDB renames the current `soltrace.out` file to `soltrace.bak`; and 3. solidDB starts a new `soltrace.out` file.	100 m Unit: 1 byte k=KB m=MB	RW/Startup
TraceSecDecimals	Number of second decimals in trace outputs. Allowed values are from 0 to 3.	3	RW/Startup

A.15 Synchronizer section

Table 64. Synchronizer parameters

[Synchronizer]	Description	Factory Value	Access Mode	Usage
ConnectStrForMaster	This parameter indicates the connection string that the master must use to communicate with the replica. This information is read when the replica server is started. The connect string is sent to the master as part of each message from the replica to the master. For example: `ConnectStrForMaster= tcp replicahost 1316`	none	RW	Replica
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	RO	Master
RefreshIsolationLevel	This parameter defines 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	Default is the same as **`SQL.IsolationLevel`**	RW	Master

Table 64. Synchronizer parameters (continued)

[Synchronizer]	Description	Factory Value	Access Mode	Usage
RefreshReadLevelRows	This parameter defines the number of rows after which the read level is released in the master, if the used isolation level is READ COMMITTED. With other isolation levels, 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. **Note:** See also the **Srv.RemoteStartTasks** parameter.	1000	RW	Master
ReplicaRefreshLoad	This parameter defines the amount of system processing capacity (as percentage) that is used to perform a refresh in the replica. By default, full capacity is used. If you want to reserve some capacity for local processing in parallel with refresh, set this parameter to a lower value. Possible values are between 0 and 100. Value 0 means that the feature is disabled and full processor capacity is used. **Note:** If this parameter is set to 0 or 100, you can set the system processing capacity with the SET SYNC parameter SYS_SYNC_REPLICA_REFRESH_LOAD.	100	RW	Replica
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:** You cannot use suffixes such as "K" or "M" to indicate Kilobytes or Megabytes. Value 0 means that no events are posted.	0	RO	Master Replica

Appendix B. Client-side configuration parameters

The client-side configuration parameters define various characteristics for usage of the solidDB ODBC client and solidDB tools such as solidDB SQL Editor (**solsql**). The client-side parameters are stored in the client-side `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 might improve performance. You can change the parameters by editing the client-side `solid.ini` configuration file.

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 runtime, they affect the connections established thereafter.

B.1 Client section

Table 65. 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 value 2 allows for prefetching 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** parameter.	decided by the server

GSKitPath	This parameter defines the path to the directory where the IBM Global Security Kit (GSKit) library is located. To connect an externally authenticated user, the client must be able to load the GSKit library from the location defined with this parameter.
	
	The value of the parameter must be a valid path. For example:
	`[Client]`
	`GSKitPath=/home/sol/soliddb-7.0/bin/`
	`[Client]`
	`GSKitPath="C:\Program Files\solidDB7.0\bin"`
	
	Tip: If the path contains a white space, enclose the path in double quotation marks.
	
	See also **Client.UseGSKit**.	

NoAssertMessages	If set to `yes`, the Windows runtime error dialog is not shown. This parameter is relevant to the Windows platform only.	no

Table 65. Client parameters (continued)

[Client]	Description	Factory Value
ODBCCharBinding	Defines the binding method for character data. The options are: • `raw` (binary) • `locale` (the current client locale is used) • `locale:<locale name>` (specific code page is used) The convention for <locale name> depends on the operating system. For example, in Linux environments, the locale name for the code page GB18030 in Chinese/China is `zh_CN.gb18030`. In Windows environments, the locale name for Latin1 code page in Finnish/Finland is `fin_fin.1252`. The value `raw` can be used when you want your database to use the binding used in the 6.3 or earlier versions of solidDB.	locale
ODBCHandleValidation	This parameter switches ODBC handle validation on or off. See also section *ODBC handle validation* in *IBM solidDB Programmer Guide* for more information about the `SQL_ATTR_HANDLE_VALIDATION` ODBC attribute.	no
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`** parameter.	decided by the server
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
UseEncryption	This parameter defines whether passwords are encrypted using DES encryption. If set to `no`, passwords are not encrypted.	yes
UseGSKit	This parameter defines whether IBM Global Security Kit (GSKit) is used for encryption of passwords for client connections. If set to `no`, passwords are not encrypted using GSKit. **Note:** To encrypt password using GSKit, the GSKit must be enabled on the client computer.	no

B.2 Communication section

Table 66. Client-side 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:** This parameter applies only to the TCP protocol.	0 (infinite)

Table 66. Client-side communication parameters (continued)

[Com]	Description	Factory Value
Connect	The **Connect** parameter defines the default network name (connect string) that the client uses to connect to the solidDB server, if the connect string is not specified in the connection parameters explicitly. This value is used also when the SQLConnect() call is issued with an empty data source name. The format of the standard solidDB connect string is: `protocol_name [options] [host_comput er_name] server_name` where `options` and `server_name` depend on the communication protocol. **Important:** In HotStandby and SMA setups, additional connect string attributes are used to specify further functionality, such as Transparent Connectivity (TC). For more details, see Network name and connect string syntax.	tcp localhost 1964 (Windows) upipe SOLID (Linux and UNIX)
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:** This parameter applies only to the TCP protocol.	OS-specific
SocketLinger	This parameter controls the TCP socket linger (SO_LINGER) behavior after a close on the socket connection is issued. It indicates if the system attempts to deliver any buffered data (`yes`), or if the system discards it (`no`), when a close() is issued.	no
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
Trace	If this parameter is set to `yes`, trace information about network messages for the established network connection is written to a file specified with the **TraceFile** parameter.	no
TraceFile	If the **Trace** parameter is set to `yes`, trace information about network messages is written to a file specified with this parameter. The trace file is output to the current working directory of the server or client, depending on which end the tracing is started.	`soltrace.out`

B.3 Data sources section

Table 67. Data Sources 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.		N/A

B.4 SharedMemoryAccess section

Table 68. Shared memory access parameters (client-side)

[SharedMemoryAccess]	Description	Factory value	Startup
SignalHandler	The **SignalHandler** parameter controls the SMA signal handler functionality. When set to yes, the SMA driver signal handler handles the signals defined with the **Signals** parameter. The SMA driver signal handler enables the SMA system to survive the most common application failures, such as killing or interrupting the applications from outside, or when one of the application threads runs within the server code, and another thread running application code causes application to crash. Upon the capture of certain signals, the signal handler closes the SMA connections safely and exits the SMA application. In most cases, the SMA server continues to run despite abnormal application exits. The SMA driver signal handler installs itself when the first SMA connection is established and uninstalls itself when the last SMA connection is closed. Previously installed signal handlers are retained.	yes	NA

Signals	This parameter defines the signals that can break the SMA connection and is handled by the SMA driver.
	
	The signals are defined as integers or with the following mnemonics: SIGSTOP, SIGKILL, SIGINT, SIGTERM, SIGQUIT, SIGABORT.
	Note: If the SMA application loops outside of the SMA driver (for example, does not call any functions), the signal can fail to terminate the application. In such a case:
	1. Throw out the connections at the server.
	`admin command 'throwout <userid>'`
	2. Use SIGKILL signal to force the SMA application to exit.
	`kill -SIGKILL <pid>`
		Linux and UNIX: SIGINT, SIGTERM
		
		Windows: SIGINT
			NA

B.5 TransparentFailover section

Table 69. TransparentFailover parameters

[TransparentFailover]	Description	Factory value
ReconnectTimeout	This parameter specifies how long (in milliseconds) the driver should wait until it tries to reconnect to the primary in case of TF switchover or failover. If the driver cannot find the new primary (reconnect), an error is returned and the TF connection becomes broken.	10000
WaitTimeout	This parameter specifies how long (in milliseconds) the driver should wait for the server to switch state. When the driver tries to reconnect to the servers, it might connect to the server being in an intermediate (switching or uncertain) state.	10000

Appendix C. solidDB command-line options

Option	Description	Examples
-c *directory*	Changes working directory	`solid -c /data/solid`
-f	Starts the server in foreground	
-m	Enables the monitoring facility for tracing SQL statements. For more details, see 5, "Monitoring solidDB," on page 91.	
-n *name*	Sets the server name	

-s install,*name,fullexepath* -c *directory*[,autostart]	The 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 a Windows service instance of the solidDB server	`solid -s"remove,SOLID"`
-s start	Specifies that solidDB starts 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"`
-U *username*	Specifies the username for the database that is being created. See also options **-x execute**, **- x executeandnoexit**, and **-x exit**.	

Option	Description	Examples
-P *password*	Specifies the password for the database that is being created. See also options **-x execute**, **- x executeandnoexit**, and **-x exit**.	
-p	Create a new database with externally authenticated database administrator	
-E	Encrypts the database. An encryption password is mandatory when **-E** is specified. The encryption password is needed to protect the symmetric encryption key which is stored in an unencrypted header page of the database file. Specify the encryption password using the **-x keypwdfile:***file_name* or **-S** *encryption_password* option.	`solid -C mycatalog -U admin -P admin123 -E -x keypwdfile:pwd.txt` `solid -C mycatalog -U admin -P admin123 -E -S admin456`
-S *encryption_password*	Specifies the database file encryption password	
-x assert:s	Disables emergency exit dialog	
-x autoconvert	Converts (migrates) the database format from a previous release version to the current release version and starts the server	
-x backupserver	Used only in HotStandby setups. Starts the server in a netcopy listening mode. A server in the netcopy listening mode accepts only netcopy operations from the Primary server.	
-C *catalog*	Specifies the database catalog name	
-x convert	Converts (migrates) database format to the current format used by solidDB and starts the server process	

-x decrypt -S *password*	Decrypts the database	`solid -x decrypt -S dba`
		
		`solid -x decrypt -x keypwdfile:pwd.txt`

-x disableallmessageboxes	Hides all message windows	
-x errormsgnostop	Does not wait for user actions on error dialogs	

-x execute: *file_name*	Prompts for the user name and password of the database administrator, creates a new database, executes SQL statements from a file, and exits.
	
	You can also use the options **-U** and **-P** to provide the DBA 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`

Option	Description	Examples

-x executeandnoexit: *file_name*	Prompts for the user name and password of the database administrator, creates a new database, executes SQL statements from a file, but does not exit.
	
	You can also use the options **-U** and **-P** to provide the DBA 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`

-x exit	Prompts for the user name and password of the database administrator, creates a new database, and exits. You can also use the options **-U** and **-P** to provide the DBA user name and password.	`solid.exe -x exit` `solid.exe -x exit -Udba -Pdba`
-x forcerecovery	Performs a forced roll-forward recovery	
-x hide	Hides the server icon	
-x ignoreerrors	Ignores index errors	
-x ignorecrashed	Ignores log files and reverts to checkpoint	
-x inifile:*file_name*	Specifies the configuration file name, instead of using the default `solid.ini` file in the working directory	
-x infodbfreefactor	Informs about unused pages The server exits after performing the task. See also:**-x reorganize**.	
-x keypwdfile: *file_name*	Reads the database encryption password from a file, instead of command line argument. This way the password cannot be seen by running the UNIX command **ps**.	
-x listen:*network_name*	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). • 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	Ignores log files during recovery	
-x pathprefix:*directory*	Uses files in the specified directory	

Option	Description	Examples
-x pwdfile: *file_name*	Reads the password from a file instead of command line argument. This way the password cannot 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.	
-x testintegrity	Performs a full database integrity test and exits	
-x testblocks	Checks the disk block integrity and produces a report in a `ssdebug.out` file. The server exits after performing the task.	
-x testindex*[:size]*	Tests database index and exits The optional *[:size]* parameter outputs index size.	
-x version	Displays the server version and exits	
-?	Help = Usage	
-h	Help = Usage	

Appendix D. Environment variables

Table 70. solidDB environment variables

Environment variable	Purpose	Example
SOLAPPINFO	Identifies applications running in the same computer and under the same username for the purposes of tracing and management SOLAPPINFO is set on the client node. The ADMIN COMMAND 'userlist' returns the value of SOLAPPINFO on the server side.The value of SOLAPPINFO must not contain blanks. **Tip:** In JDBC environments, the SOLAPPINFO can be set with the connection property solid_appinfo. Alternatively, the following Java command line may be used to pass the value of the environmental variable to the driver: `java -Dsolid_appinfo=%SOLAPPINFO% java_program_name`	`export SOLAPPINFO=testapp`
SOLIDDIR	Defines the default directory for `solid.ini` and license files	`export SOLIDDIR=/home/soliddb/settings/`
SOLSMASTART	Forces the start address space for the SMA server to the solidDB default The value depends on the operating system; see *SOLSMASTART default address spaces* in the *IBM solidDB Shared Memory Access and Linked Library Access User Guide* for more details.	`export SOLSMASTART=0x2c0000000000`
SOLTRACE	Turns on the Network trace facility, overriding the **`Com.Trace`** setting in the `solid.ini` file	`export SOLTRACE=yes`
SOLTRACEFILE	Defines the name and location of the file where trace information is output, overriding the **`Com.TraceFile`** setting in the `solid.ini` file Defining the SOLTRACEFILE environment variable automatically turns on the Network trace facility.	`export SOLTRACEFILE=/home/soliddb/settings/trace.out`

Appendix E. 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 71. 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 E.1, “solidDB system errors,” on page 279.
Database or DBE (database engine)	The errors in these classes are detected by the server and can demand administrative actions. Messages do not typically require administrative actions. For the list of errors and messages, see E.2, “solidDB database errors,” on page 282 and E.16, “solidDB DBE (database engine) errors and messages,” on page 341.
Table or TAB (table)	These errors and messages are caused by erroneous SQL statements detected by the server. Administrative actions are not needed. For the list of errors and messages, see E.3, “solidDB table errors,” on page 291 and E.26, “solidDB TAB (table) messages,” on page 349.
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 • E.5, “solidDB communication errors,” on page 307 • E.4, “solidDB session errors,” on page 306 • E.14, “solidDB COM (communication) messages,” on page 338 • E.10, “solidDB RPC errors and messages,” on page 320
Server	These errors are caused by erroneous administrative actions or client requests. They can demand administrative actions. For the list of errors, see E.6, “solidDB server errors,” on page 310
Procedure	These errors are encountered when defining or executing a stored procedure. Administrative actions are not needed. For the list of errors, see E.7, “solidDB procedure errors,” on page 316.
SA API	The SA API errors are return codes for the SA function SaSQLExecDirect. For more information, see E.8, “solidDB API errors,” on page 319 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 E.9, “solidDB sorter errors,” on page 319 and E.24, “solidDB XS (external sorter) errors and messages,” on page 348.

Table 71. solidDB error categories (continued)

Error class	Description
Synchronization or SNC	These errors can be encountered when creating or maintaining the solidDB environment. They occur when using certain solidDB statements that are proprietary SQL extensions. For the list of errors, see E.11, "solidDB synchronization errors," on page 321 and E.23, "solidDB SNC (synchronization) messages," on page 347.
HotStandby or HSB	The HotStandby errors occur when using the `ADMIN COMMAND 'HotStandby'` commands. For the list of errors, see E.12, "solidDB HotStandby errors," on page 335 and E.22, "solidDB HSB (HotStandby) errors and messages," on page 345.
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 E.13, "solidDB SSA (SQL API) errors," on page 336
CP (checkpoint)	The CP messages provide information about the status or conditions of checkpoint operations. For the list of messages, see E.17, "solidDB CP (checkpoint) messages," on page 343.
BCKP (backup)	The BCKP messages provide information about the status or conditions of backup operations. For the list of messages, see E.18, "solidDB BCKP (backup) messages," on page 343.
AT (timed commands)	The AT messages provide information about the status or conditions of executing timed commands. For the list of messages, see E.19, "solidDB AT (timed commands) messages," on page 343.
LOG (logging)	The LOG messages provide information about the status or conditions of transaction logging. For the list of messages, see E.20, "solidDB LOG (logging) messages," on page 344.
INI (configuration file)	The INI messages provide information about the use of the `solid.ini` configuration file. For the list of messages, see E.21, "solidDB INI (configuration file) messages," on page 344.
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 E.25, "solidDB FIL (file system) messages," on page 348.
SMA (shared memory access)	The SMA messages provide information about operations when solidDB is used with shared memory access. For the list of errors, see E.27, "solidDB SMA (shared memory access) errors," on page 349.
PT (passthrough)	The PT errors provide information about operations when solidDB is used with SQL passthrough. For the list of messages, see E.28, "solidDB PT (passthrough) errors," on page 349.
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 E.29, "solidDB SQL errors," on page 350
Executable errors	These errors are caused by the failure of the solidDB server 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 E.30, "solidDB executable errors," on page 356
solidDB Speed Loader (**`solloado`** or **`solload`**)	These errors are encountered when running the solidDB Speed Loader utility (**`solloado`** or **`solload`**) to load data from external files into the solidDB database. For the list of errors, see E.31, "solidDB Speed Loader (solloado and solload) errors," on page 357

In addition to the errors and messages described above, you might receive an internal error. In such a case, contact IBM Software Support at http://www.ibm.com/software/data/soliddb/support/.

E.1 solidDB system errors

Table 72. 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 **IndexFile.FileSpec** parameter.
11003	System	Fatal Error	File write failed, configuration exceeded. Writing to the database file failed because the maximum database file size set with **IndexFile.FileSpec** parameter has been exceeded. Increase the maximum file size limit or divide the database into multiple files. See 9.1.5, "Troubleshooting database file size (file write fails)," on page 199 for more details.
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, or insufficient file read or write permissions.
			
			With this error, the following type of error message can be written to the `solmsg.out` file:
			`SsBOpenLocal failed,`
			`file '/home/solid/sol00001.log',`
			`error = 13, retries = 0, open files = 1`
			
			The error 13 refers to an operating system error code 13 which is defined as:
			`#define EACCES 13 /* Permission denied */`
			
			This means that the `solid` process does not have operating system permissions to read or create the file.

11007	System	Fatal Error	File lock failure. The server failed to lock the database file.

Table 72. solidDB system errors (continued)

Code	Class	Type	Description
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 **IndexFile.FileSpec** is too long. Change the name to a proper file name.
11011	System	Error	Duplicate file name specification. Filename specified in parameter **IndexFile.FileSpec** 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.

Table 72. solidDB system errors (continued)

Code	Class	Type	Description
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.
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.

Table 72. solidDB system errors (continued)

Code	Class	Type	Description
11043	System	Fatal Error	GSKit enabled, but failed to load the GSKit library. Check the library path. If all solidDB users use external authentication and solidDB cannot load the GSKit library, the solidDB server cannot start. See also 4.6, "Troubleshooting encryption and authentication," on page 87.
11044	System	Fatal Error	External authentication requires GSKit to be enabled. If all solidDB users use external authentication and the use of IBM Global Security Kit (GSKit) is disabled (**General.UseGSKit=no**), solidDB server cannot start. See also 4.6, "Troubleshooting encryption and authentication," on page 87.
11045	System	Fatal Error	Call to system function munmap failed with errno 12 (ENOMEM). System has run out of memory, or the process's maximum number of mappings has been exceeded. In Linux environments, you might be able to recover from the error by increasing the value of the kernel parameter **vm.max_map_count**. For instructions how to modify the value of **vm.max_map_count**, see the documentation provided with your operating system.

E.2 solidDB database errors

Table 73. 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
1022	Database	Warning	The database you are using has been originally created with a different database block size setting than your current The database you are using has been originally created with a different database block size setting than your current setting. To suppress this warning, edit the `solid.ini` file to contain the following parameter setting: **Indexfile.BlockSize=<required_blocksize>**
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.

Table 73. solidDB database errors (continued)

Code	Class	Type	Description
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, and triggers.
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. The error is returned when the tables are set with optimistic concurrency control and two or more concurrent connections attempt to obtain a exclusive lock on the same row/or set of rows at the same time (same row in the database is being modified simultaneously). To diagnose the problem: 1. Enable monitoring. 2. Check `soltrace.out` for error 10006. Resolving the problem: The transaction that has been committed first is allowed to make the modifications to the database. The latter transactions is rolled back and this error message is returned to the application. To handle this update conflict, for example, the application could try to re-read the data and retry the update. You can also switch to pessimistic locking method where row-level locking is used to avoid update conflicts. The pessimistic locking mode is suggested for tables that are modified frequently. To turn the pessimistic locking on for a table, use the ALTER TABLE statement.
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.

Table 73. solidDB database errors (continued)

Code	Class	Type	Description
10010	Database	Fatal Error	No checkpoint exists in database. Possible causes for this error include: • Most likely the creation of a new database had failed. To recover, delete the database and log files and try to create the database again. • The database has been irrevocably corrupted. revert to the latest backup. 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. This can be due to a disk error or other system failure. Restore the database from a backup.
10012	Database	Fatal Error	Node split failed. This error is given if the node split of the in-memory database (B+ tree) fails.
10013	Database	Error	Transaction is read-only. You tried to do one of the following: 1) Execute conflicting SET TRANSACTION statements, for example, 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.

Table 73. solidDB database errors (continued)

Code	Class	Type	Description
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.BlockSize** 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.
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.

Table 73. solidDB database errors (continued)

Code	Class	Type	Description
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 or a lock wait timeout, transaction is rolled back. To avoid lock timeouts, adjust the lock wait timeout settings. To avoid deadlocks, adjust the data access order in concurrent transactions. If necessary, begin transaction again. For more information, see: • Lock duration and timeout • Locks and lock modes • Server timeouts
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.
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.

Table 73. solidDB database errors (continued)

Code	Class	Type	Description
10039	Database	Error	INTERNAL: Assertion failure For more information, contact IBM Software 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. Server has run out of disk space and automatically switched to read-only mode. Free disk space, move the database file to another disk or divide the database into multiple files and disks with **`IndexFile.FileSpec`** parameter. **Tip:** You can query the read-only mode of the database by using the **`ADMIN COMMAND 'getreadonlyflag'`** command.
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. 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.
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

Table 73. solidDB database errors (continued)

Code	Class	Type	Description
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. If **General.Readonly** is set to no (default), the server has run out of disk space and automatically switched to read-only mode. To free disk space, move the database file to another disk or divide the database into multiple files and disks with **IndexFile.FileSpec** parameter.

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 logfile *file_name* 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. To recover: • Remove *log filename* and all other log files with greater sequence numbers. • Change the value of the **Logging.FileNameTemplate** parameter to point to a directory that does not contain any solidDB transaction log files.
10064	Database	Fatal Error	Illegal log file name template. Most likely, the log file name template specified in **Logging.FileNameTemplate**: 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. 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.

Table 73. solidDB database errors (continued)

Code	Class	Type	Description

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. 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. Restore the database from the latest backup.
10071	Database	Fatal Error	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*. Restore the database from the latest backup.
10074	Database	Fatal Error	Roll-forward recovery failed. Revert to the latest backup.

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.

Table 73. solidDB database errors (continued)

Code	Class	Type	Description
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.
10093	Database	Fatal Error	HotStandby migration is not possible if Hotstandby is not configured.
10094	Database	Fatal Error	Only *amount* cache pages configured for M-table usage, at least *amount* 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. 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.

Table 73. solidDB database errors (continued)

Code	Class	Type	Description
10108	Database	Error	Search is aborted because of concurrent data dictionary operation on table.
10109	Database	Error	Transaction is already in prepared state, operation failed.
10110	Database	Error	XA transaction has not yet ended, operation failed.
10111	Database	Error	XA transaction has ended, operation failed.
10112	Database	Error	XA transaction is from a different connection, operation failed.
10113	Database	Error	Duplicate XID.
10114	Database	Error	XA transaction cannot have any DDL statements, operation failed.
10115	Database	Error	Operation is not supported with XA transaction.
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	Row size exceeds the allowed value for M-tables. You might receive this error, for example, after you have upgraded to a new version of solidDB that uses in-memory tables as a default table type, and the previous version of your database used disk-based tables by default. If you want to continue using disk-based tables, set the **General.DefaultStoreIsMemory** parameter to no.
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.
16509	Database	Error	MME transaction maximum size exceeded The maximum transaction size is set with the **MME.MaxTransactionSize** parameter.

E.3 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);

Error code	Class	Type	Description
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.
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.

Error code	Class	Type	Description
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.
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.

Error code	Class	Type	Description
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 '_'.
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.

Error code	Class	Type	Description
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.
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.

Error code	Class	Type	Description
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.
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.

Error code	Class	Type	Description
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.
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.

Error code	Class	Type	Description
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.
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.

Error code	Class	Type	Description
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.
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.

Error code	Class	Type	Description
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.
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`

Error code	Class	Type	Description
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.
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.

Error code	Class	Type	Description
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.
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.

Error code	Class	Type	Description
13164	Table	Error	Transaction is active, operation failed.
13165	Table	Error	Cannot 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
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.
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.

Error code	Class	Type	Description
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
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.

Error code	Class	Type	Description
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.
13206	Table	Error	Sequence <*sequence*> has incorrect START WITH constant <*constant*>
13207	Table	Error	Sequence <*sequence*> has incorrect INCREMENT BY constant <*constant*>
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.
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	Row size exceeds the allowed value 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
13451	Table	Error	Passthrough not configured. Check that you have set the **Passthrough.PassthroughEnabled** parameter to 'Yes'. Check that the SYS_SERVER table contains correct login data for the back-end.
13452	Table	Error	Passthrough backend database not available. solidDB cannot connect to the back-end data server. Check your configuration settings.

Error code	Class	Type	Description
13453	Table	Error	Passthrough cursors are forward only cursors.
13454	Table	Error	Passthrough error: <description> This error is returned to user if the back-end data server reports a failure but solidDB cannot read the actual error. The following reasons cause the this error to be output to `solmsg.out`: • The **`Passthrough.SqlPassthroughRead`** or **`Passthrough.SqlPassthroughWrite`** parameter in the `solid.ini` has an invalid value (for example, **`SqlPassthroughRead=forse`**) • The **`Passthrough.PassthroughEnabled`** parameter is set to 'Yes' but **`Passthrough.RemoteServerDriverPath`** is not defined. • The **`Passthrough.PassthroughEnabled`** parameter is set to 'Yes' but **`Passthrough.RemoteServerDSN`** is not defined. • A function in a .dll cannot be found. The error description is the function name. • A .dll cannot be found.
13455	Table	Error	Passthrough not allowed. This error is caused by violation of the set isolation level. To preserve consistency of the back-end database when using SQL passthrough, the isolation level of the front-end must be the same (or similar) or higher than in the back-end.
13456	Table	Error	Passthrough backend error: SQLState=<value>, NativeError=<back-end error identifier>, MessageText=<back-end error description>.
13457	Table	Error	Passthrough error: resultset mismatch. The table definitions in the front-end and back-end database do not match (for example, the number of columns is different).
13458	Table	Error	Passthrough error: parameter mismatch. The parameters used in an SQL statements do not match when executed in the front-end and back-end database.
13459	Table	Error	Passthrough error: Datatype is not supported.
13460	Table	Error	Server <name> already exists The back-end login data for the specified server has been created already. **Note:** The default name for the back-end data server is 'default'.
13461	Table	Error	Server <name> not found. The back-end login data for the specified server does not exist.
13463	Table	Error	Passthrough error. Distributed transaction must be read only in back-end.
13471	Table	Error	Table *table_name* does not have conditions for cache segment *cache_segment_name*
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>

E.4 solidDB session errors

Table 74. 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.

Table 74. solidDB session errors (continued)

Code	Class	Type	Description
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.
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) 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.

E.5 solidDB communication errors

Table 75. 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 *connect_string* is not supported. Most likely the protocol name in the connect string is misspelled. Check the connect string.
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.

Table 75. solidDB communication errors (continued)

Code	Class	Type	Description
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.
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.

Table 75. solidDB communication errors (continued)

Code	Class	Type	Description
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.
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.

Table 75. solidDB communication errors (continued)

Code	Class	Type	Description
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.
21330	Communication	Error	Failed to accept a new client connection, too many open files

E.6 solidDB server errors

Table 76. 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.

Table 76. solidDB server errors (continued)

Code	Class	Type	Description
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.
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.

Table 76. solidDB server errors (continued)

Code	Class	Type	Description
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.
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'**

Table 76. solidDB server errors (continued)

Code	Class	Type	Description
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'**
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 the parameter **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	Server process size has exceeded the limit set with parameter **Srv.ProcessMemoryLimit**. Only ADMIN COMMANDs are allowed. Increase the value of **Srv.ProcessMemoryLimit** or disable the process memory size checking by setting **Srv.ProcessMemoryCheckInterval** to 0. **Tip:** You can modify the **Srv.ProcessMemoryLimit** and **Srv.ProcessMemoryCheckInterval** parameters dynamically with ADMIN COMMAND 'parameter'. ADMIN COMMANDs are allowed so that you can increase the process size limit.

Table 76. solidDB server errors (continued)

Code	Class	Type	Description
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'**
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'** • **ADMIN COMMAND 'hotstandby 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).

Table 76. solidDB server errors (continued)

Code	Class	Type	Description
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.
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.

Table 76. solidDB server errors (continued)

Code	Class	Type	Description
14556	Server	Error	Netbackup: No server connection string specified.
14557	Server	Error	Netbackup: A server configured for HotStandby cannot act as a netbackup server.
14558	Server	Error	Operation not allowed when delete capture is off.
14559	Server	Error	Exceeded maximum number of allowed connections for user. Close existing connections or contact administrator to increase the limit. Users with administrative privileges can define the maximum number of connections with the ALTER USER statement.
14570	Server	Error	XID not found.
14571	Server	Error	XID has not been prepared. Cannot execute two phase commit.
14572	Server	Error	XID has been prepared. Cannot execute one phase commit.
14600	Server	Error	Command is ambiguous in cluster session.
14706	Server	Error	Invalid read thread mode for HotStandby, only mode 2 is supported.
30135	Server	Fatal Error	SMA application has failed while processing the solidDB server code. Server cannot continue and is executing emergency shutdown. Failed process id is *process_id*.

E.7 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.

Code	Class	Type	Description
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.
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.

Code	Class	Type	Description
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.

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.

Code	Class	Type	Description
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.
23530	Procedure	Error	Event wait is not allowed inside a scalar function call
23531	Procedure	Error	Event wait is not allowed inside a trigger action
23532	Procedure	Error	SQL DML statements are not allowed inside a scalar function call
23533	Procedure	Error	SQL DDL statements are not allowed inside a scalar function call
23534	Procedure	Error	External procedure/function library <name> load failed
23535	Procedure	Error	External procedure/function name <name> was not found from library <library_name>
23538	Procedure	Error	Procedure <procedure_name> output parameter number <number> assign failed
23539	Procedure	Error	Procedure/function <name> return column number <number> assign failed
23540	Procedure	Error	External procedure/function <name> reported error, sqlstate: <sqlstate> message: <message>

E.8 solidDB API errors

Table 77. 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.

E.9 solidDB sorter errors

Table 78. 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

Table 78. solidDB sorter errors (continued)

Code	Class	Type	Description
30803	Sorter	Error	Illegal value specified for parameter: <parameter>=<value>(legal range is <value>)
30804	Sorter	Error	Sorter temporary directory: <value> does not exist

E.10 solidDB RPC errors and messages

Table 79. 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.
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

Table 79. solidDB RPC errors and messages (continued)

Code	Class	Type	Description
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>

E.11 solidDB synchronization errors

Table 80. 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 **IndexFile.FileSpec** parameter setting. For details on this parameter, read "FileSpec_[1...n] parameter" on page 48. You need 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 Software Support at http://www.ibm.com/software/data/soliddb/support/.
25002	Synchronization	Error	Cannot save data dictionary statements.

Table 80. solidDB synchronization errors (continued)

Code	Class	Type	Description

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`

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`

Table 80. solidDB synchronization errors (continued)

Code	Class	Type	Description

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)`

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 again 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`
			
			For example, if the master database takes a long time to start due to, for example, a server restart, the message reply to replica can time out. To recover, reissue the MESSAGE GET REPLY command and set the timeout to a larger value or FOREVER.

Table 80. solidDB synchronization errors (continued)

Code	Class	Type	Description

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`
			
			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.
			
			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.
			
			Statements that return this error:
			`MESSAGE message_name FORWARD`
			`MESSAGE message_name GET REPLY`
			`MESSAGE message_name RESTART`

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. Statements that return this error: `ALTER TABLE table_name SET SYNCHISTORY`

Table 80. solidDB synchronization errors (continued)

Code	Class	Type	Description

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`
			
			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 about registering a replica database.
			
			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`
			
			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`

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).
			
			Statements that return this error:
			`CREATE PUBLICATION publication_name ...`
			`DROP PUBLICATION publication_name REGISTRATION`
			`SET SYNC MAINTENANCE MODE ...;`
			`ALTER TABLE table_name SET SYNCHISTORY`

Table 80. solidDB synchronization errors (continued)

Code	Class	Type	Description

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 the server 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.
			
			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.
			
			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. The server 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:** This error can be returned if you use double quotation marks rather than single quotation marks around the *master_connect_string* in a MESSAGE FORWARD command.
			
			Statements that return this error:
			`IMPORT 'filename'`
			`MESSAGE message_name FORWARD TO 'master_connect_string'`
			`TIMEOUT timeout_in_seconds`
			`MESSAGE message_name GET REPLY ...`
			`MESSAGE message_name APPEND REFRESH publication_name`
			`MESSAGE message_name EXECUTE ....`

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`
			
			Statements that return this error:
			`DROP PUBLICATION publication_name REGISTRATION`
			`MESSAGE message_name APPEND REGISTER REPLICA`
			`MESSAGE message_name BEGIN ...`

Table 80. solidDB synchronization errors (continued)

Code	Class	Type	Description

25026	Synchronization	Error	A user who has not been defined in the master database, attempts to perform a solidDB SQL command.
			
			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, you have two options:
			• 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`
			• Add an appropriate user to the master database, and download it by executing the following command:
			`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. 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.
			
			Statements that return this error:
			`MESSAGE message_name FORWARD ...`
			`MESSAGE message_name GET REPLY ...`

25031	Synchronization	Error	Transaction is active, operation failed.
			
			A replica attempts to process a message when having an active transaction.
			
			Statements that return this error:
			`IMPORT 'filename'`
			`MESSAGE message_name FORWARD ...`
			`MESSAGE message_name GET REPLY TIMEOUT ...`
			`MESSAGE message_name EXECUTE`

Table 80. solidDB synchronization errors (continued)

Code	Class	Type	Description
25032	Synchronization	Error	All publication SQL statements must return rows. The publication definition contains SQL operations that do not return rows. Only SELECT statements are allowed in the publication. 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. 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. Statements that return this error: `MESSAGE message_name BEGIN`

25035	Synchronization	Error	Message *message_name* is in use.
			
			A 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 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.
			
			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 ...`

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.
			
			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.
			
			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`

Table 80. solidDB synchronization errors (continued)

Code	Class	Type	Description

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.
			
			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. 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.
			
			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.
			
			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`

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. 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. Statements that return this error: `MESSAGE message_name GET REPLY`

Table 80. solidDB synchronization errors (continued)

Code	Class	Type	Description
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. 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 server that is 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.
			
			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. Statements that return this error: `CREATE PUBLICATION publication_name ...`
25050	Synchronization	Error	Table has no history.
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. Statements that return this error: `SET SYNC REPLICA NO`
25052	Synchronization	Error	Failed to set node name to *node_name*. The *node_name* can be invalid.
25053	Synchronization	Error	Replica not registered in master.

Table 80. solidDB synchronization errors (continued)

Code	Class	Type	Description

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.
			
			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. Statements that return this error: `MESSAGE message_name FORWARD TO connect_info options`

25056	Synchronization	Error	Autocommit not allowed.
			
			The statement must be executed with autocommit mode turned off.
			
			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.
			
			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.
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. Statements that return this error: `SET SYNC NODE NAME unique_node_name`

Table 80. solidDB synchronization errors (continued)

Code	Class	Type	Description

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.
			
			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. 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. 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. 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. Statements that return this error: `DROP REPLICA replica_name`
25065	Synchronization	Error	Unfinished message *message_name* found for master *master_name*. Dropping the master failed because there are unfinished messages. 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. Statements that return this error: `CREATE SYNC BOOKMARK`

Table 80. solidDB synchronization errors (continued)

Code	Class	Type	Description
25067	Synchronization	Error	Synchronization bookmark *bookmark_name* not found. Bookmark name is not an existing bookmark. 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. Statements that return this error: `EXPORT SUBSCRIPTION`
25069	Synchronization	Error	Import file *file_name* open failure. Failed to open import file for IMPORT. 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. 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.
			Statements that return this error:
			`DROP PUBLICATION publication_name REGISTRATION`
			`MESSAGE message_name APPEND REFRESH publication_name`

25072	Synchronization	Error	Already registered to publication *publication_name*. Replica is already registered to a publication. Statements that return this error: `MESSAGE message_name APPEND REGISTER REPLICA`
25073	Synchronization	Error	Export file can have data only from one master.

Table 80. solidDB synchronization errors (continued)

Code	Class	Type	Description

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.
			
			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. Statements that return this error: `SET SYNC NODE NONE`
25083	Synchronization	Error	Commit block cannot be used with HotStandby.
25084	Synchronization	Error	Cannot 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.
25086	Synchronization	Error	Cannot save START statement.

Table 80. solidDB synchronization errors (continued)

Code	Class	Type	Description

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. If you have not defined the connect string in the replica solid.ini file, registering a replica does not automatically add the connect string into the SYS_SYNC_REPLICAS table. Define the connection information in the following way:
			`[Synchronizer]`
			`ConnectStrForMaster=<connect_string>`
			
			For example:
			`[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, you do not receive 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.

E.12 solidDB HotStandby errors

Table 81. 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.

Table 81. solidDB HotStandby errors (continued)

Code	Class	Type	Description
14701	HotStandby	Error	Rejected connection, both servers in SECONDARY role. Meaning: Command **'hsb connect'** returns this error if both nodes are in same role.
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.

E.13 solidDB SSA (SQL API) errors

Table 82. 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.

Table 82. solidDB SSA (SQL API) errors (continued)

Error code			Description
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.
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).

Table 82. solidDB SSA (SQL API) errors (continued)

Error code			Description
25216	SSA	Error	Connection switch, some session context may be lost This error is specific to HotStandby configurations that use transparent connectivity (TF-1). A TF-1 connection has encountered a connection switch. The application must roll back the transaction to restore the connection.
25217	SSA	Error	Client unable to establish a primary connection This error is specific to HotStandby configurations that use transparent connectivity (TF-1). 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

E.14 solidDB COM (communication) messages

Table 83. 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.

Table 83. solidDB COM (communication) messages (continued)

Code	Class	Type	Description
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.
30022	COM	Message	User <username> failed to connect database, character set is utf8, unsupported by client.
30023	COM	Message	User <username> failed to connect, default char binding uses codepage <codepage> Your client does not support the Unicode database mode; update the client to the same version as the server.
30024	COM	Message	Failed to set **Com.ListenThreadPriority**, value not changed. **Note: Com.ListenThreadPriority** is an internal parameter. IBM Software Support might ask you to set this parameter for troubleshooting purposes.

E.15 solidDB SRV (server) errors

Table 84. 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...
30106	SRV	Message	Failed to set **Srv.TaskThreadPriority**, value not changed. **Note: Srv.TaskThreadPriority** is an internal parameter. IBM Software Support might ask you to set this parameter for troubleshooting purposes.
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>

Table 84. solidDB SRV errors (continued)

Code	Class	Type	Description
30128	SRV	Message	Using license file <license_file>
30129	SRV	Message	Signal <value>
30130	SRV	Message	Server 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 backup server 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.
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.
30158	SRV	Message	Parameter **General.MultiprocessingLevel** has been set automatically to <value>, the number of logical CPUs detected. As of V6.5 Fix Pack 4, the factory value of **General.MultiprocessingLevel** is read from the system as the number of logical processing units. With some processor architectures, the number of logical processing units might not be the same as the number of physical cores. In such cases, the optimal value for this parameter typically varies between the number of the physical cores and the number of logical processing units.
30159	SRV	Message	Failed to load external authentication module. Cannot login externally authenticated users. See 4.6, “Troubleshooting encryption and authentication,” on page 87 for more details.
30160	SRV	Message	GSKit is not enabled. Cannot login externally authenticated users. See 4.6, “Troubleshooting encryption and authentication,” on page 87 for more details.
30161	SRV	Message	Strong encryption enabled.
30162	SRV	Message	Strong encryption disabled, using default.
30163	SRV	Message	Opening an encrypted database file.
30164	SRV	Message	Encrypting the database file
30165	SRV	Message	Decrypting the database file.
30166	SRV	Message	Database file is not encrypted.

E.16 solidDB DBE (database engine) errors and messages

Table 85. 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. 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.
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, 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

Table 85. solidDB DBE errors and messages (continued)

Code	Class	Type	Description
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 in the `solid.ini` were found. 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 re-create 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.
30262	DBE	Message	Maximum number of users reached.
30264	DBE	Message	Fixing bad B-tree node reference at address *address*.
30265	DBE	Message	Fixing B-tree key range errors at address *address*.
30266	DBE	Message	Running autofix.
30267	DBE	Message	Autofix completed successfully.
30268	DBE	Message	Autofix failed, check `ssdebug.log` for errors.
30269	DBE	Message	Maximum number of logreader operations reached.
30270	DBE	Message	Maximum number of abort operations reached.
30270	DBE	Message	Failed to set **`Logging.ThreadPriority`**, value not changed.
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
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

E.17 solidDB CP (checkpoint) messages

Table 86. 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 prolonged, <number> of <number> pages left.
30287	CP	Message	Checkpoint MME flushing prolonged, <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.

E.18 solidDB BCKP (backup) messages

Table 87. 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`.

E.19 solidDB AT (timed commands) messages

Table 88. solidDB AT (timed commands) messages

Code	Class	Type	Description
30350	AT	Message	At: backup <backup_directory>
30351	AT	Message	At: makecp
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, restart the server.
30360	AT	Message	AT command failed. <reason>

Table 88. solidDB AT (timed commands) messages (continued)

Code	Class	Type	Description
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

E.20 solidDB LOG (logging) messages

Table 89. 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.

E.21 solidDB INI (configuration file) messages

Table 90. solidDB INI (configuration file) messages

Code	Class	Type	Description
30157	INI	Message	Parameter <parameter> is incompatible with SMA server and is thus ignored.
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>
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	**Srv.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 process memory size checking by setting **Srv.ProcessMemoryLimit** to 0.

E.22 solidDB HSB (HotStandby) errors and messages

Table 91. 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
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. Restart secondary and execute a HSB netcopy.
30550	HSB	Message	Connection broken to HotStandby secondary server

Table 91. solidDB HSB errors and messages (continued)

Code	Class	Type	Description
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*
30561	HSB	Message	Started connecting to HotStandby
30562	HSB	Message	Connection broken to HotStandby primary server
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.
30586	HSB	Message	HotStandby catchup, *catchup_phase* logpos: *log_position* *catchup_phase* can be: • HSB waitdurable • HSB catchup start • HSB write catchup • HSB write switch
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.
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*.

Table 91. solidDB HSB errors and messages (continued)

Code	Class	Type	Description
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.
30793	HSB	Message	Maximum number of secondary tasks *value* reached. The queue at the secondary server for incoming log operations is growing faster than the operations can be executed and acknowledged to the primary server. The queue can be monitored with the performance counter **HSB secondary queues**.
30794	HSB	Message	Invalid **HotStandby.Connect** option `-z`. `-z` option is not supported.

E.23 solidDB SNC (synchronization) messages

Table 92. 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

E.24 solidDB XS (external sorter) errors and messages

Table 93. 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.
30805	XS	Message	Maximum number of files for external sorting reached

E.25 solidDB FIL (file system) messages

Table 94. 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>

Table 94. solidDB FIL (file system) messages (continued)

Code	Class	Type	Description
30924	FIL	Message	File write seek failed, error <error_code>, file <file_name>, location <directory> retries <value>, vaxc$error = <value>
30925	FIL	Message	File write end failed, error <error_code>, file <file_name>, retries <value>
30926	FIL	Message	File write end failed, error <error_code>, file <file_name>, retries <value>, vaxc$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>

E.26 solidDB TAB (table) messages

Table 95. 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>

E.27 solidDB SMA (shared memory access) errors

Table 96. solidDB SMA (shared memory access) errors

Code	Class	Type	Description
31100	SMA	Fatal Error	Value for maximum shared memory size **SharedMemoryAccess.MaxSharedMemorySize=<value>** is invalid.

E.28 solidDB PT (passthrough) errors

Table 97. solidDB passthrough errors

Code	Class	Type	Description
32001	PT	Error	Passthrough: <description>
32002	PT	Error	Passthrough: Error:<description>

E.29 solidDB SQL errors

Table 98. 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.
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.

Table 98. solidDB SQL errors (continued)

Error code	Description
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.
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.

Table 98. solidDB SQL errors (continued)

Error code	Description
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.
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.

Table 98. solidDB SQL errors (continued)

Error code	Description
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.
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 ? < ?;

Table 98. solidDB SQL errors (continued)

Error code	Description
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.
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.

Table 98. solidDB SQL errors (continued)

Error code	Description
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.
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`.

Table 98. solidDB SQL errors (continued)

Error code	Description
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
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.

E.30 solidDB executable errors

Table 99. 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

Table 99. solidDB executable errors (continued)

Error code	Description
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
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.

E.31 solidDB Speed Loader (solloado and solload) errors

Table 100. solidDB Speed Loader (solloado and 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

Table 100. solidDB Speed Loader (solloado and solload) errors (continued)

Error Code	Meaning
116	Communication error, connection is lost
121	RPC parameter error
122	Table not found
124	Wrong number of parameters

Appendix F. solidDB ADMIN COMMAND syntax

This appendix describes the solidDB ADMIN COMMAND syntax. This command set is not part of ANSI SQL; it is an extension that is specific to the solidDB product.

F.1 ADMIN COMMAND

```
ADMIN COMMAND 'command_name'

command_name ::= ABORT | ASSERTEXIT | BACKUP |
BACKGROUNDJOB | BACKUPLIST | CHECKPOINTING | CLEANBGJOBINFO |
CLOSE | DESCRIBE | ERRORCODE | ERROREXIT | ERRORMESSAGE | FILESPEC |
GETREADONLYFLAG | HELP | HOTSTANDBY | INDEXUSAGE | INFO | LOGMESSAGE |
LOGREADER | MAKECP | MEMORY | MESSAGES | MONITOR | NETBACKUP |
NETBACKUPLIST | NETSTAT | NOTIFY | OPEN | PARAMETER | PASSTHROUGH STATUS |
PERFMON | PERFMON DIFF | PERFMON TIMERS | PID | PROCTRACE | PROTOCOLS |
REPORT | RUNMERGE | SAVE | SHUTDOWN | SQLLIST | STARTMERGE | STATUS |
THROWOUT | TID | TRACE | TRACEMESSAGE | USERID | USERLIST |
USERTRACE | VERSION
```

Usage

The ADMIN COMMAND is a SQL extension specific to solidDB server. You use ADMIN COMMANDs to execute administrative operations.

Using ADMIN COMMAND with solidDB SQL Editor (solsql)

When used with the solidDB SQL Editor (**solsql**), the *command_name* must be given with single quotation marks. For example:

```
ADMIN COMMAND 'backup'
```

If you use double quotation marks, the *command_name* is not recognized and the command fails.

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 quotation marks. For example:

```
backup
```

Abbreviations

Abbreviations for ADMIN COMMANDs are also available. For example:

```
ADMIN COMMAND 'bak'
```

To access a list of abbreviated commands, execute the following command:

```
ADMIN COMMAND 'help'
```

Return values

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.

Help

To access a list of abbreviated commands, execute the following command:

```
ADMIN COMMAND 'help'
```

To access the options and syntax description for a specific command, execute the following command:

```
ADMIN COMMAND 'command_name help'
```

Important:

- **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. The concurrency control in solidDB 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.

Table 101. 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 might 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.

Table 101. ADMIN COMMAND syntax and options (continued)

Option syntax	Description

`ADMIN COMMAND 'backgroundjob' [LIST [-l] [user]] | [ABORT {jobid | user | ALL }] | [DELETE ERRORINFO {jobid | user | ALL }]'`
 	
`user ::= USER {username|userid}`
 	
`Abbreviation: bgjob`
	Lists and optionally 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 in a synchronized or an asynchronic (default) manner. The synchronized operation is specified by using the **`-s`** option. The default backup directory is defined with the **`General.BackupDirectory`**. The backup directory can 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.
`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.

Table 101. ADMIN COMMAND syntax and options (continued)

Option syntax	Description
`ADMIN COMMAND 'errormessage <string>'` `Abbreviation: errmsg`	Outputs the user-defined `<string>` to the error message log (`solerror.out`).

`ADMIN COMMAND 'filespec [-d | -a "<file_name> <max_file_size_in_bytes> [<device_number>]"]'`
`Abbreviation: fs`
	Displays or modifies database (index) file specifications defined with the **`IndexFile.FileSpec`** parameter as well file sizes and current fill ratios (percentage).
	• **`-d`** deletes the database file specified with `<file_name max_file_size_in_bytes> [device_number]`
	• **`-a`** adds a new database file specification as specified with `<file_name> >max_file_size_in_bytes> [<device_number>]`
	For example:
	`ADMIN COMMAND 'fs -a "solid3.db 3000M"';`
	`RC TEXT`
	`-- ----`
	`0 Added: FileSpec_3 = solid3.db 3145728000`
	The database file specification changes are stored in the `solid.ini` configuration file at shutdown.

`ADMIN COMMAND 'getreadonlyflag'` `Abbreviation: grof`	Returns the read-only status of the database. You can set the database set to read-only mode with the **`General.Readonly`** parameter. Alternatively, if the server runs out of disk space, it switches to read-only mode automatically.
`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*.
`ADMIN COMMAND 'indexusage'` `Abbreviation: idxu`	Displays the indexes, showing the number of times each index has been used.

Table 101. 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 (KB).
	• logsize - Size of log files (KB).
	• uptime - Server startup timestamp.
	• bcktime - Timestamp of last successfully completed local backup.
	• cptime - Timestamp of last successfully completed checkpoint.
	• tracestate - Current trace state - see `ADMIN COMMAND 'trace'` for information on tracing.
	• 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 state for accepting connections. `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 allocated memory (bytes).
	• dbfreesize - Amount of free space remaining in database (KB).
	• dbpagesize - Database page size (KB).
	• 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. You can set the server name with the solidDB startup option `-n name`.
	• primarystarttime - The time the Primary role has started.
	• secondarystarttime- The time the Secondary role has started.
	• dbconfigsize - The configured database size (MB), as set with the **`IndexFile.FileSpec`** parameter.
	• dbcreatetime | dbcreationtime - Database creation timestamp.
	• processsize | psize - System-level virtual process size (KB).
	
	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 101. ADMIN COMMAND syntax and options (continued)

Option syntax	Description
`ADMIN COMMAND 'logreader stop [all|<partition_id>]'` `Abbreviation: lr`	This command stops the transmission of log records on active log reader connections. When this command is issued, the active log reader applications reach the end of the result set (`SQLSTATE 0200, No data found`) when fetching the next row of the SYS_LOG table. If the form LOGREADER STOP or LOGREADER STOP ALL is used, all log record transmissions are stopped. If a <PARTITION_ID> is given, the command affect only the log reader operation on that partition. To access the log again, the application needs to reconnect. The log reading may be resumed without any loss of information if the last read position is known. If the SYS_LOG table is accessed without specifying the log position, the reading starts from the live data. **Important:** The stopping of the log transmission is effective immediately, regardless of the fact that there might be records in the log awaiting transmission. If the server is running in the relaxed durability mode (default), do not execute LOGREADER STOP before all the records are written to the log, if those records are meant to be seen in the log reader. With the default logging settings, it is safe to wait for 5 seconds after the last write operation.
`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, that is, the amount of memory allocated by the server based on internal solidDB memory counters, including the memory used by data in the in-memory tables. **Note:** 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. Optionally, you can also specify the severity and message numbers of the output. 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.

Table 101. ADMIN COMMAND syntax and options (continued)

Option syntax	Description
`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. *options* can be • `-s` Synchronized execution • `-I` Executes a full database integrity check • `-i` Executes a database index integrity check `DELETE_LOGS | KEEP_LOGS` defines whether backup logs are deleted or kept in the source server. Default is `DELETE_LOGS`. **Note:** • `DELETE_LOGS` is referred to as *Full backup*. • `KEEP_LOGS` is referred to as *Copy backup*. Using `KEEP_LOGS` corresponds to setting the **`General.NetbackupDeleteLog`** parameter to `no`. `connect connect str` specifies the connection to the NetBackup Server. If *connect str* is omitted, it must be specified in the `solid.ini` configuration file. For the full connect string syntax, see Format of the connect string. `dir backup dir` defines the backup directory in the NetBackup Server. The path can be either absolute or relative to the **`netbackup`** root directory. 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 `solid.ini` 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 101. ADMIN COMMAND syntax and options (continued)

Option syntax	Description
`ADMIN COMMAND 'parameter [-r] [-t] [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. • **`-t`** means that the changed value is not stored in the `solid.ini` file (same as **`temporary`**). • **`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 'passthrough status' Abbreviation: pt`	Provides the following status information about the SQL passthrough connections: • NO REMOTE SERVER - no remote server object defined • NOT CONNECTED - not connected, no errors • CONNECTED - connected • LOGIN FAILED - failed at login • CONNECTION BROKEN - connection broken

Table 101. ADMIN COMMAND syntax and options (continued)

Option syntax	Description
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'**
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'**

Table 101. ADMIN COMMAND syntax and options (continued)

Option syntax	Description

`ADMIN COMMAND 'perfmon timers [ start | stop | list | clear ]'`
 	
`Abbreviation: pmon timers`
	Produces information about execution times of database operations such as SQL execute and file operations for each user.
	
	`start` starts the timers and clears the existing counter values.
	
	`stop` stops the timers and keeps the current counter values.
	
	`list` lists the current counter values.
	
	`clear` clears the current counter values.
	
	The timer information is given in seconds. The values are cumulative since last **`perfmon timers start`** or **`perfmon timers clear`**.
	
	The output can be viewed in the console window (**`perfmon timers list`**) or printed into a report file with **`ADMIN COMMAND 'report report_name'`**. In the report file, the timer information is listed under the section PERFORMANCE TIMERS. The output lists the execution times for each user, identified with the userid.
	
	For more information and examples of the output, see ADMIN COMMAND 'perform timers'.

`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.
	
	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 and statistics 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 the server process. If the force option is used, the active transactions are aborted and the users are disconnected forcefully.

Table 101. ADMIN COMMAND syntax and options (continued)

Option syntax	Description
`ADMIN COMMAND 'sqllist [top number_of_statements]'`	Prints out a list of the longest running SQL statements among the currently running statements. You must specify the number of statements you want to list.
`ADMIN COMMAND 'startmerge'` `Abbrevation: sm`	Starts and waits for completion of merge.
`ADMIN COMMAND 'status'` `Abbreviation: sta`	Displays statistics for the server since the startup. The output provides the following information: • Server startup timestamp • Working directory • Configuration file location and name • Memory statistics – Amount of memory allocated by the server based on internal solidDB memory counters, including the memory used by data in the in-memory tables (same as ADMIN COMMAND 'memory' output value) • Process size statistics in KB – Resident set size - Actual process size in memory as reported by the operating system – Virtual size - System-level virtual process size (same as ADMIN COMMAND 'info processsize' output value) • Transaction count statistics: – Commit – Number of committed transactions – Abort – Number of system-aborted transactions – Rollback – Number of transactions rolled back by user – Total – Total number of committed, aborted, and rolled back transactions – Read-only – Number of read-only transactions – Trxbuf – Number of transactions in transaction buffer – Active – Number of active transactions (same as performance counter *Trans active*) – Validate – Number of active transactions being validated at commit phase (same as performance counter *Trans validate*) • Cache count statistics: – Hit rate – Percentage of successful bufferpool cache hits (disk access avoided) – Find – Number of searches in cache – Read – Number of read operations on disk – Write – Number of write operations from cache to disk • Database statistics: – Index writes – Number of write operations – (Index writes) After last merge – Number of write operations since last merge – Log writes – Number of log write operations – (Log writes) After last cp – Number of log write operations since last checkpoint – Active searches – Number of active searches on database engine level – (Active searches) Average – Average number of active searches on database engine level – Database size – Log size • User count statistics – Current – Number of current connected users – Maximum – Number of concurrently connected users since startup – Total – Number of connected users since startup For more information, see Checking database status.

Table 101. ADMIN COMMAND syntax and options (continued)

Option syntax	Description
`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 | passthrough | xa | hac | info <level> | func | proc | 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. • passthrough - provides tracing information about the SQL passthrough connections and the loading of the ODBC driver as follows: – Loading of the ODBC driver: driver name and status of the load – Status of connections to the back-end: connect/reconnect/disconnect/broken • xa - distributed transaction information • hac - High Availability Controller (HAC); trace information is output to `hactrace.out` in the HAC working directory **Note:** To start tracing on HAC, you must issue the command on a HAC connection. For example, connect to HAC with **`solsql`** using the port defined with the **`HAController.Listen`** parameter in the `solidhac.ini` configuration file. • info <level> - SQL execution trace (level can be 0...8) • func - function execution information • proc - stored procedure execution information • all - both SQL messages and network communications messages are written to the trace file. • active - lists all active traces

Table 101. ADMIN COMMAND syntax and options (continued)

Option syntax	Description
`ADMIN COMMAND 'tracemessage <string>'` `Abbreviation: trcmsg`	Outputs the user-defined `<string>` to the trace message log (`soltrace.out`).

`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 101. 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*, *Tid*, *Type*, *Machine Id*, *Login time*, *Client version*, and *Appinfo* (if available). With the `-l` option, the following information is displayed: • *User name* - The user name of the connected user. • *User Id* - The user session identification number (userid) within the database. The lifetime of the userid is that of the user session. After the user logs out, the number can be reused. • *Tid* - The identification number as a 4-digit code of the current user thread in the server. • *Type* - Client type. Possible values are: – *Java*, which refers to a client using JDBC – *ODBC*, which refers to a client using ODBC, including solidDB SQL Editor (**`solsql`**) – *Solcon*, which refers to solidDB Remote Control (**`solcon`**) • *Machine id* - The client computer name (host name) and its IP address, if available • *Login time* - The client computer login timestamp • *Client version* - The version of the JDBC or ODBC client, as of V7.0.0.2 Interim Fix 2. **Note:** – The client version information is not available for clients prior to V7.0.0.2 Interim Fix 2 or for solidDB Remote Control (**`solcon`**) connections. – For solidDB SQL Editor (**`solsql`**) connections, the ODBC client version is shown. • *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.

Table 101. ADMIN COMMAND syntax and options (continued)

Option syntax	Description

..continued..
`ADMIN COMMAND 'userlist [-l] [name | id]'`
`Abbreviation: ul`
	• *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.
	• *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* - The timestamp of the most recent statement.
	• *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 101. 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

-x autoconvert (command line option) 271
-x convert (command line option) 271
@
  AT (@) sign 137

A

abnormal shutdown 32
AbortTimeOut (parameter) 251
AdaptiveRowsPerMessage (parameter) 251
ADMIN COMMAND
  abort 360
  assertexit 360
  backgroundjob 361
  backup 361
  backuplist 361
  checkpointing 361
  cleanbgjobinfo 361
  close 361
  commands 359
  describe 361
  errorcode 361
  errorexit 361
  filespec 362
  getreadonlyflag 362
  help 362
  hotstandby 362
  indexusage 362
  info 363
  info processsize 167
  logreader 364
  makecp 364
  memory 364
  messages 364
  monitor 364
  netbackup 365
  netbackuplist 365
  netstat 365
  notify 365
  open 365
  parameter 366
  passthrough status 366
  perfmon 367
  perfmon diff 367
  perfmon timers 368
  pid 368
  proctrace 368
  protocols 368
  runmerge 368
  save parameters 368
  shutdown 368
  sqllist 369
  startmerge 369
  status 369
  syntax 359
  throwout 370
  tid 370
  trace 370
  userid 371
ADMIN COMMAND *(continued)*
  userlist 372, 373
  usertrace 374
  version 374
ADMIN COMMAND 'perfmon'
  server performance 98
ADMIN COMMAND 'report report_filename'
  producing a report for troubleshooting 97
ADMIN COMMAND 'status backup'
  querying last backup status 97
ADMIN COMMAND 'status'
  querying database status 95
ADMIN COMMAND 'throwout' 12
  disconnecting users 97
ADMIN COMMAND 'userlist'
  querying for connected users 96
administering
  multiple servers manually 5
AllowConnect (parameter) 252
AllowDuplicateIndex (parameter) 247
ANSI (reserved word) 144
at commands 34
AuditTrailEnabled (parameter) 86, 247
autocommit 177
autoconvert
  command line option 271
automating administrative tasks 5, 34

B

backup 28
  automating 34
  configuring and automating 27
  failed 31
  local 24
  manual 24
  monitoring and controlling 30
  network backup 25
  network backup, server administration 30
  querying 97
  restoring 31
  timed commands 34
  typical problems 31
BackupBlockSize (parameter) 217
BackupCopyIniFile (parameter) 217
BackupCopyLog (parameter) 217
BackupCopySolmsgOut (parameter) 217
BackupDeleteLog (parameter) 217
BackupDirectory (parameter) 217
BackupFlushInterval (parameter) 217
BackupStepsToSkip (parameter) 218
bcktime ADMIN COMMAND 363
BLANKS
  solidDB Speed Loader 144
BLOBs (Binary Large Objects) 19
  defining 19
BlockSize (parameter) 18, 231, 234, 246
Bonsai Tree 175, 177
BonsaitreeJoinLimit (parameter) 232
BtreeJoinLimit (parameter) 232

C

cache (disk-based) 169
CacheSize (parameter) 49, 232
CAST (function) 318
catalogs
  name criteria 13
CHARACTERSET keyword (solload) 146
CharPadding (parameter) 247
checkpoint
  'makecp' command 364
CheckpointDeleteLog (parameter) 218
CheckpointInterval (parameter) 176, 218
checkpoints 33
  automatic daemon 33
  automating 34
  erasing automatically 33
  forcing 176
  frequency 176
  timed commands 34
  tuning 176
client-side configuration parameters 267
ClientReadTimeout (parameter) 268
closing solidDB 12
  ADMIN COMMAND 12
columns
  setting LONG VARCHAR 19
command line options 271
COMMIT WORK statement
  application code 178
  troubleshooting 178
communication
  between client and server 117
  selecting a protocol 125
  tracing problems 190
communication protocols 125
  Named Pipes 127
  selecting 125
  summary 128
  supported protocols 125
  TCP/IP 125
  UNIX Pipes 127
communication tracing 52
configuration file
  description 17
  server-side 38
  setting 42
  solidDB Speed Loader 139
configuring
  client-side configuration file 38
  configuration file 38
  default settings 38
  example 38
  factory values 38
  managing parameters 38, 39, 40
  parameter settings 38
  server-side configuration file 38
  setting parameters 39, 41
  solid.ini 38
  viewing parameter descriptions 40
  viewing parameters 38, 39
Connect (parameter) 52, 269
connect string 52
  clients 121
connecting to solidDB
  basics 20
  login 20
ConnectionCheckInterval (parameter) 252
connections
  committed transactions 178
  determining existing 178
ConnectStrForMaster (parameter) 264, 335
ConnectTimeOut (parameter) 252, 269
control file (solidDB Speed Loader)
  description 138
  syntax 144
convert
  command line option 271
converting database format 271
ConvertOrsToUnionsCount (parameter) 248
counters 99
cptime ADMIN COMMAND 363
creating
  checkpoints 33
CursorCloseAtTransEnd (parameter) 248

D

database
  automating 34
  backing up 24
  block size 18
  cache 169, 170
    changing dynamically 170
    size 169
  checking last backup status 97
  checking overall status 95
  closing 11, 34
  compacting 35
  configuring 38
  converting format 271
  creating 13
  creation time 363
  currently connected users 96
  decreasing database file size 48
  defining objects 19
  disconnecting a user 97
  file size
    decreasing 48
  free space in 363
  in-memory 38
  index file 48
  location 18, 47
  login 57
  maximum size 18
  monitoring 98
  opening 34
  performance 98
  querying last backup 97
  recovery 32
  restoring master and replica 24
  several databases on one computer 23
  shrinking 35, 271
  shutting down 12
  size 13, 47
  troubleshooting 98
database mode
  partial Unicode 15
  Unicode 15
DatabaseSizeReportInterval (parameter) 253
DataDictionaryErrorMaxWait (parameter) 219
DATE data type
  Speed Loader 146
dbconfigsize ADMIN COMMAND 363
dbcreatetime ADMIN COMMAND 363

dbfreesize ADMIN COMMAND 363
dbpagesize ADMIN COMMAND 363
dbsize ADMIN COMMAND 363
DecFloatPrecision16 (parameter) 248
DecimalPrecAsNumeric (parameter) 219
decrypting databases 81
DefaultDomainName (parameter) 219
DefaultStoreIsMemory (parameter) 219
DES encryption 76
DigitTemplateChar (parameter) 234
DirectIO (parameter) 232, 234
DisableIdleMerge (parameter) 219
DisableOutput (parameter) 92, 253
disconnecting users 12
durability
  relaxed 163
  strict 163
DurabilityLevel (parameter) 235

E

Echo (parameter) 253
EmulateOldTimestampDiff (parameter) 248
EnableHints (parameter) 248
ENCLOSURE (solidDB Speed Loader) 148
encryption
  DES
    changing password 80
    database 78
    decrypting 81
    password 80
    starting encrypted database 80
  disabling 81
  GSKit
    database 78
  level 82
entering timed commands 34
environment variables
  SOLTRACE 190
  SOLTRACEFILE 190
error codes
  error handling 277
error handling
  AT messages 343
  BCKP messages 343
  COM messages 338
  communication errors 307
  CP messages 343
  database errors 282
  DBE errors 341
  error codes 277
  executable errors 356
  FIL messages 348
  HotStandby errors 335
  HSB errors 345
  INI messages 344
  LOG messages 344
  passthrough errors 349
  procedure errors 316
  RPC errors 320
  SA API errors 319
  server errors 310
  SMA errors 349
  SNC errors 347
  sorter errors 319
  Speed Loader errors 357
  SQL API errors 336
error handling *(continued)*
  SQL errors 350
  SRV errors 339
  synchronization errors 321
  system errors 279
  TAB messages 349
  table errors 291
  XS errors 348
events
  soldd and listing event definitions 157
ExecRowsPerMessage (parameter) 253, 267
ExecuteNodataODBC3Behaviour (parameter) 248
executing
  system commands, automated 34
ExtendIncrement (parameter) 175, 232
external sorting 171

F

file locations 16
file system 16
FileFlush (parameter) 235
FileNameTemplate (parameter) 51, 236
FileSpec (parameter) 18, 48
FileWriteFlushMode (parameter) 219
ForceThreadsToSystemScope (parameter) 254
free space in database 363

G

GSKit encryption 76
GSKitLoginRequired (parameter) 220
GSKitPath (parameter) 220, 267

H

HealthCheckEnabled (parameter) 254
HealthCheckInterval (parameter) 254
HealthCheckTimeout (parameter) 254

I

I/O
  distributing 175
  tuning 175
IBM Global Security Kit (GSKit 76
IBMPC (reserved word) 146
IgnoreOnDisabled (parameter) 243
ImdbMemoryLimit (parameter) 240
ImdbMemoryLowPercentage (parameter) 240
ImdbMemoryWarningPercentage (parameter) 240
imdbsize ADMIN COMMAND 363
ImplicitStart (parameter) 213
import file (solidDB Speed Loader) 139
index file
  splitting to multiple disks 48
Info (parameter) 249
InfoFileFlush (parameter) 249
InfoFileName (parameter) 249
InfoFileSize (parameter) 249
InifileLineSplitting 255
InternalCharEncoding (parameter) 221
INTO_TABLE_PART
  solidDB Speed Loader 148
IOThreads (parameter) 221

isolation levels
read committed 165
repeatable read 165
serializable 165
IsolationLevel (parameter) 249

K

KeepAllOutFiles (parameter) 255

L

Latin1CaseSemantics (parameter) 249
Listen (parameter) 214
listen name 118, 120, 121
listing users 373
local backup 24
LocalStartTasks (parameter) 255
LockEscalationEnabled (parameter) 241
LockEscalationLimit (parameter) 241
LockHashSize (parameter) 222, 241
LockWaitTimeOut (parameter) 223
log files
overview 32
solerror.out 92
solmsg.out 92
Speed Loader 139
LogDir (parameter) 236
LogEnabled (parameter) 236
logging
transaction durability 163
transactions 32
logical data source names
defining in solid.ini 123
login
description 57
incorrect username or password 57
LogReaderEnabled (parameter) 238
logsize ADMIN COMMAND 363
LogSoftMemoryLimit (parameter) 236
LogWriteMode (parameter) 236
LongSequentialSearchLimit (parameter) 223

M

makecp 176
manual administration 5
master database
backing up 24
restoring 24
MasterStatementCache (parameter) 264
MaxBgTaskInterval (parameter) 256
MaxBlobExpressionSize (parameter) 19, 250
MaxBytesCachedInPrivateMemoryPool (parameter) 241
MaxCacheUsage (parameter) 241
MaxCacheUsePercent (parameter) 246
MaxConstraintLength (parameter) 257
MaxFilesTotal (parameter) 246
MaxLogSize (parameter) 238
MaxMemLogSize (parameter) 238, 239
MaxMemPerSort (parameter) 246
MaxMergeParts (parameter) 223
MaxMergeTasks (parameter) 223
MaxNestedProcedures (parameter) 250
MaxNestedtriggers (parameter) 250
MaxOpencursors (parameter) 257
MaxOpenFiles (parameter) 223
MaxPhysMsgLen (parameter) 214
MaxRPCDataLen (parameter) 257
MaxSharedMemorySize (parameter) 245
MaxSpace (parameter) 238, 239
MaxStartStatements (parameter) 257
MaxTransactionSize (parameter) 242
MaxUsers (parameter) 257
maxusers ADMIN COMMAND 363
MaxWriteConcurrency (parameter) 223
memory
physical 169
tuning 166
virtual 169
MemoryPoolScope (parameter) 242
MemoryReportDelta (parameter) 257
MemoryReportLimit (parameter) 258
MemorySizeReportInterval (parameter) 258
memtotal ADMIN COMMAND 363
MergeInterval (parameter) 175, 224
message log 92
MessageLogSize (parameter) 258
MinCheckpointTime (parameter) 176, 224
MinMergeTime (parameter) 224
MinSplitSize (parameter) 237
monitoring 91
monitorstate ADMIN COMMAND 363
MSWINDOWS (reserved word) 144
MultiprocessingLevel (parameter) 224

N

Name (parameter) 258
name ADMIN COMMAND 363
Named Pipes 127
netbackup 25
NetBackupConnect (parameter) 224
NetBackupConnectTimeout (parameter) 224
NetBackupCopyIniFile (parameter) 224
NetBackupCopyLog (parameter) 224
NetBackupCopySolmsgOut (parameter) 225
NetBackupDeleteLog (parameter) 225
NetBackupDirectory (parameter) 225
NetBackupDirectory (parameters) 50
NetBackupReadTimeout (parameter) 225
NetBackupReceiveBufferSize (parameter) 225
NetBackupRootDir (parameter) 258
network backup
directory 50
overview 25
network communication
tracing 52
troubleshooting 210
network messages
tuning 175
network names 118, 120, 121
adding 120
clients 121
defining 47, 52
modifying 120
Named Pipes 127
removing 121
TCP/IP 125
UNIX Pipes 127
viewing 120
network trace facility 190
nmp 127

nmpipe 127
NoAssertMessages (parameter) 267
non-graphical user interfaces
  creating new database 13
NULLIF
  Speed Loader 144, 151
NULLSTR
  solidDB Speed Loader 144
NumberOfMemoryPools (parameter) 242
numcursors ADMIN COMMAND 363
NumericPadding (parameter) 250
numlocks ADMIN COMMAND 363
nummerges ADMIN COMMAND 363
numtransactions ADMIN COMMAND 363
numusers ADMIN COMMAND 363

O

ODBC
  Connect parameter 52
  connect string 52
ODBCCharBinding (parameter) 268
ODBCDefaultCharBinding (parameter) 259
ODBCHandleValidation (parameter) 268
open ADMIN COMMAND 12
openstate ADMIN COMMAND 363
operating system
  tuning 169

P

PAMServiceName (parameter) 225
parameters 213
  BlockSize 18
  CacheSize 49
  CheckpointInterval 176
  client-side 267
  Connect 52
  ExtendIncrement 175
  FileNameTemplate 51
  FileSpec 18, 48
  format 46
  MaxBlobExpressionSize 19
  MergeInterval 175
  MinCheckpointTime 176
  NetBackupDirectory 50
  ProcessMemoryCheckInterval 167, 168
  ProcessMemoryLimit 167, 168
  ProcessMemoryLowPercentage 168
  ProcessMemoryWarningPercentage 168
  setting 175
  SortArraySize 171
  Threads 51
  Trace 52, 54
  TraceFile 52
partial Unicode database mode 15
passthrough
  errors 349
PassthroughEnabled (parameter) 243
passwords
  administrator 14
  changing 14
  criteria 13
  maximum number of characters 13
  reset 14
PCOEM (reserved word) 144
performance
  counters 99
  diagnosing problems 179
  snapshot of 98
  tuning 163, 179
performing batch mode operations 5
Pessimistic (parameter) 225
PessimisticTableUseNFetch (parameter) 259
phantom 165, 166
  updates
    repeatable read 165
    serializable 166
physical memory 169
ping facility 192
POSITION
  solidDB Speed Loader 151
PreferExactNumericFunctions (parameter) 250
PreFlushPercent (parameter) 233
PRESERVE BLANKS
  solidDB Speed Loader 148
primarystarttime ADMIN COMMAND 363
PrintMsgCode (parameter) 92, 259
problem determination
  troubleshooting 181
procedure errors 316
ProcedureCache (parameter) 251
process size
  controlling 167
  elements 167
ProcessMemoryCheckInterval (parameter) 167, 168, 260
ProcessMemoryHysteresisPercentage (parameter) 260
ProcessMemoryLimit (parameter) 167, 168, 260
ProcessMemoryLowPercentage (parameter) 168, 261
ProcessMemoryWarningPercentage (parameter) 168, 261
processsize ADMIN COMMAND 363
psize ADMIN COMMAND 363

Q

querying database
  ADMIN COMMAND 'status' 95

R

RConnectLifetime (parameter) 214
RConnectPoolSize (parameter) 215
RConnectRPCTimeout (parameter) 215
READ COMMITTED 264
ReadAhead (parameter) 233
ReadBufSize (parameter) 215
ReadLevelMaxTime (parameter) 225
ReadMostlyLoadPercentAtPrimary (parameter) 214
Readonly (parameter) 226
ReadThreadMode (parameter) 261
recovery 163
  automatic roll-forward 24
ReferenceCacheSizeForHash (parameter) 234
RefreshIsolationLevel (parameter) 264
RefreshReadLevelRows (parameter) 265
relaxed durability 163
RelaxedMaxDelay (parameter) 237
ReleaseMemoryAtShutdown (parameter) 242
RemoteServerDriverPath (parameter) 243
RemoteServerDSN (parameter) 243
RemoteStartTasks (parameter) 262
REPEATABLE READ 264

replica databases
  backing up 24
  restoring 24
ReplicaRefreshLoad (parameter) 265
ReportInterval (parameter) 262
reports
  automating 34
  creating a continuous performance monitoring report 99
  creating a report for troubleshooting 97
  creating a status report 97
  full list of perfmon counters 102
RestoreThreads (parameter) 243
Restoring backups 31
roles
  _SYSTEM 84
  database administration 84
  PUBLIC 84
  SYS_ADMIN_ROLE 84
  SYS_CONSOLE_ROLE 84
  SYS_SYNC_ADMIN_ROLE 84
  SYS_SYNC_REGISTER_ROLE 84
roll-forward recovery 24
RowsPerMessage (parameter) 262, 268
RpcEventThresholdByteCount (parameter) 265
RSAKeySize (parameter) 226
running several servers 23

S

SCAND7BIT (reserved word) 144
scripts
  calling 137
  executing SQL script from file 137
SearchBufferLimit (parameter) 226
secondarystarttime ADMIN COMMAND 363
sernum ADMIN COMMAND 363
server errors 310
server names
  network names 118
server-side configuration parameters 213
SharedMemoryAccessRights (parameter) 245
shortcut (Windows)
  server 11
  solsql 11
shutdown 12
Silent (parameter) 239, 262
SimpleOptimizerRules (parameter) 251
SocketLinger (parameter) 215, 269
SocketLingerTime (parameter) 215, 269
soldd 155
solerror.out
  description 92
solexp 152
solid.ini
  configuration parameters 213, 267
  configuring solidDB 37
  description 17
solidDB
  administering solidDB 5
  command line options 271
  connecting to 20
  executable program 6
  starting 6
solidDB AT messages 343
solidDB BCKP messages 343
solidDB Bonsai Tree 177
  reducing size 177
solidDB COM (communication) messages 338
solidDB communication errors 307
solidDB CP messages 343
solidDB Data Dictionary 155
  starting 155
solidDB data management tools
  overview 131
  solcon 131
  soldd 131
  solexp 131
  solload 131
solidDB database errors 282
solidDB DBE errors 341
solidDB executable
  -x execute command line option 160
  command line options 271
  errors 356
solidDB Export 152
  starting 152
solidDB FIL messages 348
solidDB HotStandby errors 335
solidDB HSB errors 345
solidDB INI messages 344
solidDB JDBC Driver
  troubleshooting 209
solidDB LOG messages 344
solidDB ODBC Driver
  troubleshooting 208
solidDB Remote Control (solcon) 131
  commands 133
  starting 132
solidDB RPC errors 320
solidDB SA API errors 319
solidDB server shortcut (Windows) 11
solidDB session errors 306
solidDB SMA errors 349
solidDB SNC errors 347
solidDB Speed Loader
  control file 138
  control file syntax 144
  description 138
  errors 357
  import file 139
  ini file 139
  log file 139
solidDB SQL
  errors 350
  troubleshooting 208
solidDB SQL API Errors 336
solidDB SQL Editor
  executing SQL statements 136
  starting 134
solidDB SQL Editor (solsql) 134
solidDB SQL Editor (solsql) shortcut (Windows) 11
solidDB SRV errors 310, 339
solidDB TAB messages 349
solidDB XS errors 348
solload 138, 140
solloado 138, 140
solmsg.out 20
  description 92
SolmsgBackupFileNum (parameter) 262
solsql 134
SOLTRACE environment variable 190
SOLTRACEFILE environment variable 190
SortArraySize (parameter) 171, 251
sorter errors 319

SorterEnabled (parameter) 247
sorting 171
space ADMIN COMMAND 363
SQL trace level
  setting 51
SQLInfo (parameter) 251
SqlPassthroughRead (parameter) 243
SqlPassthroughWrite (parameter) 243
StackTraceEnabled (parameter) 263
StandardDateTimeFormat (parameter) 263
starting
  solidDB 6
  solidDB Remote Control 132
  solidDB SQL Editor 134
StartupForceMerge (parameter) 226
StatementCache (parameter) 268
StatementMemoryTraceLimit (parameter) 263
strict durability 163
supported protocols 120
synchronization errors 321
SynchronizedWrite (parameter) 234
SyncWrite (parameter) 237
syntax
  ADMIN COMMAND 359
SYS_ADMIN_ROLE
  for database administration 84
SYS_AUDIT_TRAIL 87
SYS_CONSOLE_ROLE 84
SYS_R_MAXBYTES_IN (parameter)
  description 329
SYS_R_MAXBYTES_OUT (parameter)
  message length 329
SYS_SYNC_ADMIN_ROLE
  for database administration 84
SYS_SYNC_REGISTER_ROLE
  for database administration 84
system errors 279

T

table errors 291
TableLockWaitTimeout (parameter) 226
TCP/IP 125
TcpKeepAlive (parameter) 215
TcpKeepAliveIdleTime (parameter) 215
TcpKeepAliveProbeCount (parameter) 216
TcpKeepAliveProbeInterval (parameter) 216
TERMINATION
  solidDB Speed Loader 150
thread
  setting for processing 51
ThreadPriority (parameter) 237
Threads (parameter) 51, 263
throwing out users
  automating 34
throwout 97
throwout all 12
TIME
  solidDB Speed Loader 146
timed commands 34
  and backups 34
  and checkpoints 34
  at 34
TIMESTAMP
  solidDB Speed Loader 146
TimestampDisplaySize19 (parameter) 251
TmpDir_[1... N] (parameter) 247
Trace (parameter) 52, 54, 216, 269
trace files 93
  description 92
TraceBackupFileNum (parameter) 263
TraceFile (parameter) 52, 216, 269
TraceLogSize (parameter) 264
TraceSecDecimals (parameter) 264
tracestate ADMIN COMMAND 363
tracing communication 190
Tracing Failed Login Attempts 93
transaction logging
  overview 32
  overwriting 32
  ping-pong 32
TransactionEarlyValidate (parameter) 227
TransactionHashSize (parameter) 227
transactions
  committing to reduce Bonsai Tree size 177
  logging 32
  logs
    specifying directory 50
TriggerCache (parameter) 251
troubleshooting
  systematic problem solving 181
tuning
  checkpoints 176
  I/O 175
  memory allocation 166
  network messages 175
  operating system 169

U

Unicode
  database mode 15
UNIX Pipes 127
UpCaseQuotedIdentifiers (parameter) 251
uptime ADMIN COMMAND 363
UseEncryption (parameter) 227, 268
UseGSKit (parameter) 227, 268
user roles
  administrator 84
  system console role 84
userlist ADMIN COMMAND 372, 373
usernames
  criteria 13
  default 13
  maximum number of characters 13
users
  throwing out 34
UseThrottling (parameter) 239

V

VersionedPessimisticReadCommitted (parameter) 227
VersionedPessimisticRepeatableRead (parameter) 228
virtual memory 169

W

Windows shortcuts 11
working directory 16
WriteBufSize (parameter) 216
WriterIOThreads (parameter) 228

Notices

© Copyright Oy IBM Finland Ab 1993, 2013.

All rights reserved.

No portion of this product may be used in any way except as expressly authorized in writing by IBM.

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 is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. 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.

SC27-3839-05