



solidDB **Administrator Guide**

Version 6.0 | June 2007

solidDB Administration Guide

Copyright © 2007 Solid Information Technology Ltd.

Document number: SAG60

Product version: 06.00.1018

Date: 19.06.2007

All rights reserved. No portion of this product may be used in any way except as expressly authorized in writing by Solid Information Technology.

Solid logo with the text "SOLID" or "Solid", or "solidDB" is a registered trademark of Solid Information Technology Inc.

Solid AcceleratorLib™, Solid Availability™, Solid Bonsai Tree™, Solid BoostEngine™, Solid CarrierGrade Option™, Solid Database Engine™, Solid Diskless™, Solid EmbeddedEngine™, Solid FlowControl™, Solid FlowEngine™, Solid High Availability™, Solid HotStandby™, Solid Information Technology™, Solid Intelligent Transaction™, Solid Remote Control™, Solid SmartFlow™, Solid SQL Editor™, Solid SynchroNet™, and Built Solid™ are trademarks of Solid Information Technology Inc. All other products, services, companies and publications are trademarks or registered trademarks of their respective owners.

This product is protected by U.S. patents 6144941, 6970876, and 6978386.

This product contains the skeleton output parser for bison ("Bison"). Copyright (c) 1984, 1989, 1990 Bob Corbett and Richard Stallman.

Bison is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 1, or (at your option) any later version. Bison is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

For a period of three (3) years from the date of this license, Solid Information Technology Inc. will provide you, the licensee, with a copy of the Bison source code upon receipt of your written request and the payment of Solid's reasonable costs for providing such copy.

This product contains lexical analyzer Flex. Copyright (c) 1990 The Regents of the University of California. All rights reserved.

This code is derived from software contributed to Berkeley by Vern Paxson. The United States Government has rights in this work pursuant to contract no. DE-AC03-76SF00098 between the United States Department of Energy and the University of California. Redistribution and use in source and binary forms are permitted provided that: (1) source distributions retain this entire copyright notice and comment, and (2) distributions including binaries display the following acknowledgement: "This product includes software developed by the University of California, Berkeley and its contributors" in the documentation or other materials provided with the distribution and in all advertising materials mentioning features or use of this software. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

This product contains zlib general purpose compression library version 1.1.4, March 11th, 2002. Copyright (C) 1995-2002 Jean-loup Gailly and Mark Adler.

This software is provided "as-is", without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software. Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following

restrictions: 1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required. 2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software. 3. This notice may not be removed or altered from any source distribution.

This product is assigned the U.S. Export Control Classification Number ECCN=5D992a.

Table of Contents

1 Welcome	1
1.1 About This Guide	1
1.1.1 Organization	1
1.1.2 Audience	2
1.2 Conventions	2
1.2.1 About solidDB	2
1.2.2 Typographic Conventions	2
1.2.3 Syntax Notation	3
1.3 Solid Documentation	4
2 Managing Data with solidDB	7
2.1 solidDB Data Management Components	7
2.1.1 Programming Interfaces (ODBC and JDBC)	7
2.1.2 Network Communications Layer	8
2.1.3 SQL Parser and Optimizer	8
2.1.4 solidDB	9
2.1.5 System Tools and Utilities	9
2.2 solidDB Architecture	10
2.2.1 Data Storage	11
2.2.2 Solid SQL Optimizer	12
2.2.3 Solid Network Services	14
2.2.4 Multithread Processing	14
3 Administering solidDB	17
3.1 What You Should Know	17
3.1.1 Installing solidDB	17
3.1.2 Using Solid Embedded Engine Databases 2.20 or Prior	17
3.1.3 Special Roles for Database Administration	17
3.1.4 Automated and Manual Administration	18
3.2 Starting solidDB	19
3.3 Creating a New Database	19
3.4 Login	21
3.5 About solidDB Databases	21
3.5.1 Solid Configuration File (solid.ini)	21
3.5.2 Setting Database BlockSize and Location	21
3.5.3 Defining Database Objects	22
3.6 Connecting to solidDB	23
3.7 Viewing the solidDB Message Log	25
3.7.1 Disabling Message Log Output	25
3.7.2 Using Trace Files	26
3.7.3 Enabling Message Codes	26

3.7.4 Tracing Failed Login Attempts	26
3.8 Monitoring solidDB	27
3.8.1 Checking Overall Database Status	27
3.8.2 Obtaining Currently Connected Users	29
3.8.3 Throwing Out a Connected solidDB User	30
3.8.4 Querying the Status of the Most Recent Backup	30
3.8.5 Detailed DBMS Monitoring (Perfmon)	31
3.8.6 Producing a Status Report	37
3.9 Shutting Down solidDB	37
3.10 Performing Backup and Recovery	38
3.10.1 Making Local Backups	39
3.10.2 Making Backups Over Network	40
3.10.3 Configuring and Automating Backups	42
3.10.4 What Happens During Backup?	43
3.10.5 Administering Network Backup Server	45
3.10.6 Monitoring and Controlling Backups	45
3.10.7 Correcting a Failed Backup	46
3.10.8 Typical Problems in Backups	46
3.10.9 Restoring Backups	47
3.10.10 Transaction Logging	48
3.11 Creating Checkpoints	49
3.12 Closing a Database	50
3.13 Running Several Servers on One Computer	50
3.14 Entering Timed Commands	51
3.14.1 To Enter a Timed Command Manually	51
3.14.2 To Enter a Timed Command in SolidConsole	51
3.15 Compacting the Database Files	52
3.15.1 What Is Database Reorganization?	52
3.15.2 How Does the Database Reorganization Work?	52
3.15.3 Database Reorganization Command Line Options	52
3.16 Encrypting a Database	53
3.16.1 Enabling Encryption	53
3.16.2 Protecting the Encryption Key	53
3.16.3 Creating an Encrypted Database	54
3.16.4 Starting an Encrypted Database	54
3.16.5 Changing the Encryption Key Password	54
3.16.6 Decrypting a Database	55
3.16.7 Encryption Query	55
3.16.8 Backups	56
3.16.9 HSB	56
3.16.10 Accelerator	56
3.16.11 Performance Impact	57

4 Configuring solidDB	59
4.1 Configuration Files and Parameter Settings	59
4.2 Most Important Client-Side Parameters	60
4.2.1 Defining Network Names (Com section)	60
4.3 Most Important Server-Side Parameters	62
4.3.1 Defining Network Names (Com section)	62
4.3.2 Managing Database Files and Caching (<i>IndexFile</i> section)	63
4.3.3 Specifying the Local Backup Directory (General Section)	65
4.3.4 Specifying the Network Backup Directory (General section)	66
4.3.5 Specifying a Directory for the External Sorter Algorithm (Sorter Section)	68
4.3.6 Setting Threads for Processing (Srv Section)	68
4.3.7 Setting SQL Trace Level (SQL Section)	69
4.3.8 Specifying Network Communication Tracing (Com Section)	69
4.4 Managing Server-Side Parameters	70
4.4.1 Viewing and Setting Parameters with ADMIN COMMAND	70
4.4.2 Viewing and Setting Parameters in <i>solid.ini</i>	74
4.4.3 Constant Parameter Values	74
5 Using Solid Data Management Tools	75
5.1 Entering Password from a File	75
5.2 Solid Remote Control (solcon)	76
5.2.1 Starting Solid Remote Control	76
5.2.2 Entering Commands in Solid Remote Control	77
5.3 Solid SQL Editor (solsql)	78
5.3.1 Starting Solid SQL Editor	78
5.3.2 Executing SQL Statements with Solid SQL Editor	80
5.4 SolidConsole	81
5.4.1 Starting SolidConsole	82
5.4.2 SolidConsole Interface Features	83
5.5 Solid SpeedLoader	84
5.5.1 Control File	85
5.5.2 Import File	85
5.5.3 Message Log File	86
5.5.4 Configuration File	86
5.5.5 Starting Solid SpeedLoader	87
5.5.6 Loading Fixed-Format Records	100
5.5.7 Loading Variable-Length Records	101
5.5.8 Running a Sample Load Using Solload	103
5.5.9 Hints to Speed up Loading	103
5.6 Solid Export	104
5.6.1 Starting Solid Export	104
5.7 Solid Data Dictionary	106

5.7.1 Starting Solid Data Dictionary	106
5.8 Tools Sample: Reloading a Database	108
5.8.1 To Reload the Database	108
6 Performance Tuning	111
6.1 Logging and Transaction Durability	111
6.1.1 Background	112
6.1.2 Balancing Performance and Safety	113
6.1.3 How Relaxed Transaction Durability Can Improve Performance	113
6.1.4 Standards Compliance	114
6.1.5 Limitations on Transaction Durability	114
6.2 Choosing Transaction Isolation Levels	114
6.2.1 Setting the Isolation Level	115
6.3 Understanding Memory Consumption	116
6.3.1 Tuning Your Operating System	117
6.3.2 Database Cache	117
6.3.3 Sorting	118
6.3.4 Using In-Memory Database	119
6.4 Tuning Network Messages	120
6.5 Tuning I/O	120
6.5.1 Distributing I/O	120
6.5.2 Setting the MergeInterval Parameter	121
6.6 Tuning Checkpoints	122
6.7 Reducing Bonsai Tree Size by Committing Transactions	123
6.7.1 Preventing Excessive Bonsai Tree Growth	123
6.8 Diagnosing Poor Performance	125
7 Managing Network Connections	129
7.1 Communication between Client and Server	129
7.2 Managing Network Names	129
7.2.1 Viewing Supported Protocols for the Server	130
7.2.2 Viewing Network Names for the Server	131
7.2.3 Adding and Modifying a Network Name for the Server	131
7.2.4 To Remove a Network Name from the Server	132
7.2.5 Factory Value for a Network Name	132
7.3 Connect Strings for Clients	133
7.3.1 Mapping Logical Data Source Names to Connect Strings	134
7.3.2 Default Connect String	135
7.4 Communication Protocols	135
7.4.1 Shared Memory	136
7.4.2 TCP/IP	136
7.4.3 UNIX Pipes	138
7.4.4 Named Pipes	139
7.4.5 NetBIOS	140

7.4.6 DECnet	141
7.4.7 A Summary of Protocols	141
7.5 Logical Data Source Names	142
8 Diagnostics and Troubleshooting	145
8.1 Tracing Communication between Client and Server	145
8.1.1 The Network Trace Facility	145
8.1.2 The Ping Facility	148
8.2 Problem Reporting	150
8.3 Problem Categories	150
8.3.1 Solid ODBC API Problems	151
8.3.2 Solid ODBC Driver Problems	151
8.3.3 Solid JDBC Driver Problems	152
8.3.4 UNIFACE Driver for solidDB Problems	152
8.3.5 Communication between a Client and Server	152
8.3.6 Database Disk Block Integrity	153
A Server-Side Configuration Parameters	155
A.1 Setting Parameters through the <code>solid.ini</code> Configuration File	155
A.1.1 Rules for Formatting the <code>solid.ini</code> File	156
A.2 Changing Parameters through an ADMIN COMMAND	161
A.3 Descriptions of Configuration Parameters	162
A.4 Accelerator Section	163
A.5 Cluster Section	164
A.6 Communication Section	164
A.7 General Section	167
A.8 HotStandby Section	177
A.9 IndexFile Section	182
A.10 Logging Section	185
A.11 MME Section	189
A.12 Sorter Section	194
A.13 SQL Section	195
A.14 Srv Section	199
A.15 Synchronizer Section	208
B Client-Side Configuration Parameters	211
B.1 Setting Client-Side Parameters through the <code>solid.ini</code> Configuration File	211
B.1.1 Rules for Formatting the Client-Side <code>solid.ini</code> File	211
B.2 Descriptions of Client-Side Configuration Parameters	212
B.3 Communication Section	212
B.4 Data Sources	213
B.5 Client	214
C solidDB Command Line Options	215
D Error Codes	221

D.1 Error Categories	221
D.2 Solid Synchronization Errors	222
D.3 Solid SQL Errors	245
D.4 Solid SQL API Errors	253
D.5 Solid Database Errors	255
D.6 Solid Executable Errors	266
D.7 Solid System Errors	267
D.8 Solid Table Errors	270
D.9 Solid Server Errors	287
D.10 Solid Communication Errors	290
D.11 Solid Communication Warnings	294
D.12 Solid Procedure Errors	294
D.13 Solid Sorter Errors	298
D.14 Solid SpeedLoader Utility (solload) Errors	298
E solidDB ADMIN COMMAND Syntax	301
E.1 ADMIN COMMAND	301
E.1.1 Supported in	301
E.1.2 Usage	301
Glossary	319
Index	333

List of Figures

2.1 solidDB Components	9
3.1 SolidConsole: Status	29

List of Tables

1.1 Typographic Conventions	2
1.2 Syntax Notation Conventions	3
3.1 Starting the Server	19
3.2 Connecting to solidDB	24
3.3 Perfmon Counters	33
3.4 Options for the backup Command	39
3.5 Options for the netbackup Command	40
3.6 Parameter Correspondence to the <code>solid.ini</code> File for Local Backup	42
3.7 Parameter Correspondence to the <code>solid.ini</code> File for Netbackup	42
3.8 Available Backup and Netbackup Commands	46
3.9 Arguments and Defaults for Different Timed Commands	52
4.1 Connect String Options	61
5.1 solcon Command Options	77
5.2 Remote Control Specific Commands	78
5.3 solsql Command Options	79
5.4 solconsole Command Options	82
5.5 solload Command Options	87
5.6 SpeedLoader Reserved Words	88
5.7 Full Syntax of the Control File	89
5.8 Data Masks	92
5.9 solexp Command Options	105
5.10 soldd Command Options	106
6.1 Determining Command Status	124
6.2 Determining which Connections Have Committed Transactions	124
6.3 Diagnosing Poor Performance	126
7.1 Connect String Format	133
7.2 Format Used in the <code>solid.ini</code> File	136
7.3 Format Used in the <code>solid.ini</code> File	136
7.4 Format Used in the <code>solid.ini</code> File	138
7.5 Format Used in the <code>solid.ini</code> File	139
7.6 Format Used in the <code>solid.ini</code> File	140
7.7 Format Used in the <code>solid.ini</code> File	141
7.8 solidDB Protocols and Network Names	141
7.9 Application Protocols and Network Names	142
8.1 Ping Facility Levels	149
A.1 Accelerator Parameters	163
A.2 Cluster Parameters	164
A.3 Communication Parameters	164
A.4 General Parameters	167

A.5 HotStandby Parameters	177
A.6 IndexFile Parameters	182
A.7 Logging Parameters	185
A.8 MME parameters	189
A.9 Sorter Parameters	194
A.10 SQL Parameters	195
A.11 Srv Parameters	199
A.12 Synchronizer Parameters	208
B.1 Communication Parameters	212
B.2 Data Source Parameters	213
B.3 Client Parameters	214
C.1 solidDB Command Line Options	215
D.1 solidDB Error Categories	221
D.2 Solid Synchronization Errors	222
D.3 Solid SQL Errors	245
D.4 Solid SQL API Errors	253
D.5 Solid Database Errors	255
D.6 Solid Executable Errors	266
D.7 Solid System Errors	267
D.8 Solid Table Errors	270
D.9 Solid Server Errors	287
D.10 Solid Communication Errors	290
D.11 Solid Communication Warnings	294
D.12 Solid Procedure Errors	294
D.13 Solid Sorter Errors	298
D.14 Solid SpeedLoader Utility (solload) Errors	298
E.1 ADMIN COMMAND Syntax	303

List of Examples

3.1 Query the List of All Completed Backups and Their Success Status	46
3.2 Abort an Active Network Backup Operation	46
5.1 Remote Control	77
5.2 SQL Script Examples	80
5.3 Date Example in Control File	92
5.4 Date, Time, and Timestamp Examples in Control File	93
5.5 Treatment of Enclosed Characters within the Data	96
5.6 Treatment of Final Enclosing Character	96
5.7 Embedding Newline Characters	97
5.8 Using NULLIF Keyword with Keyword BLANKS	99
5.9 Using NULLIF Keyword with Keyword BLANKS	100
5.10 Control File Example 1	100
5.11 Control File Example 2	101
5.12 Control File Example 3	101
5.13 Control File Example 4	102
5.14 Solid Data Dictionary Examples	108
5.15 Reload the Database: Walkthrough	109
8.1 Defining Parameter Trace in the Client-Side Configuration File	146
8.2 Defining Environment Variables	146
8.3 Using Network Name Options	147
8.4 Network Trace Facility Output	147
8.5 Running Ping Facility at Level 1	149
8.6 How the Listen Parameter Restricts the Use of Ping Facility	149
B.1 Client-Side solid.ini File	212

Chapter 1. Welcome

Solid's technology, a unique combination of relational database management and advanced synchronization, is easily embeddable into any network element — from today's wireless terminals and optical routers to tomorrow's smart appliances. With solidDB, data is available anytime, anywhere, on any device.

1.1 About This Guide

solidDB Administration Guide describes how to set up, use, and administer solidDB. It also contains some information on how to optimize performance of the engine.

1.1.1 Organization

This guide is divided into the following chapters:

- Chapter 2, *Managing Data with solidDB*, familiarizes you with the underlying components of solidDB.
- Chapter 3, *Administering solidDB*, covers the typical administration tasks such as starting, connecting to, and shutting down servers. It also explains how to perform routine maintenance such as creating backups and checkpoints, and using timed commands. In addition, it shows you how to maintain your synchronization environment.
- Chapter 4, *Configuring solidDB*, describes how to set solidDB parameters for customization to meet your own environment, performance, and operations needs.
- Chapter 5, *Using Solid Data Management Tools*, describes the available utilities for performing database administration tasks, specifying SQL commands and queries, and performing specific database operations, such as loading and unloading databases.
- Chapter 6, *Performance Tuning*, describes how to optimize solidDB to improve performance.
- Chapter 7, *Managing Network Connections*, describes how to connect to solidDB using different communication protocols.
- Chapter 8, *Diagnostics and Troubleshooting*, describes tools to use for observing and tracing performance problems.
- The Appendixes give you detailed information about configuration parameters, command line options, and error messages.
- The Glossary provides definitions of solidDB terminology.

1.1.2 Audience

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

1.2 Conventions

1.2.1 About solidDB

solidDB from Solid Information Technology (Solid) represents a family of advanced database solutions for mission-critical applications.

This documentation assumes that all options of solidDB are licensed for use. In some cases, however, a customer may choose not to license certain options. These include in-memory engine, disk-based engine, CarrierGrade Option (also known as "HotStandby" in previous releases), and SmartFlow Option. Please refer to your organization's contract with Solid, or contact your Solid account representative.

1.2.2 Typographic Conventions

This manual uses the following typographic conventions:

Table 1.1. 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.
<code>solid.ini</code>	These fonts indicate file names and path expressions.
<code>SET SYNC MASTER YES; COMMIT WORK;</code>	This font is used for program code and program output. Example SQL statements also use this font.
<code>run.sh</code>	This font is used for sample command lines.
<code>TRIG_COUNT()</code>	This font is used for function names.
<code>java.sql.Connection</code>	This font is used for interface names.
<code>LockHashSize</code>	This font is used for parameter names, function arguments, and Windows registry entries.

Format	Used for
<i>argument</i>	Words emphasised like this indicate information that the user or the application must provide.
<i>solidDB Administration Guide</i>	This style is used for references to other documents, or chapters in the same document. New terms and emphasised issues are also written like this.
File path presentation	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.

1.2.3 Syntax Notation

This manual uses the following syntax notation conventions:

Table 1.2. Syntax Notation Conventions

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

Format	Used for
<ul style="list-style-type: none">..	

1.3 Solid Documentation

Below is a complete list of documents available for solidDB. Solid documentation is distributed in an electronic format, usually PDF files and web pages.

Electronic Documentation

- *Release Notes*. This file contains installation instructions and the most up-to-date information about the specific product version. This file (`releasenotes.txt`) is copied onto your system when you install the software.
- *solidDB Getting Started Guide*. This manual gives you an introduction to the solidDB.
- *solidDB SQL Guide*. This manual describes the SQL commands that solidDB supports. This manual also describes some of the system tables, system views, system stored procedures, etc. that the engine makes available to you. This manual contains some basic tutorial material on SQL for those readers who are not already familiar with SQL. Note that some specialized material is covered in other manuals. For example, the Solid "administrative commands" related to the High Availability (HotStandby) Option are described in the *solidDB High Availability User Guide*, not the solidDB SQL Guide.
- *solidDB Administration Guide*. This guide describes administrative procedures for solidDB servers. This manual includes configuration information. Note that some administrative commands use an SQL-like syntax and are documented in the *solidDB SQL Guide*.
- *solidDB Programmer Guide*. This guide explains in detail how to use features such as Solid Stored Procedure Language, triggers, events, and sequences. It also describes the interfaces (APIs and drivers) available for accessing solidDB and how to use them with a solidDB database.
- *solidDB In-Memory Database User Guide*. This manual describes how to use the in-memory database of solidDB In-memory Engine.
- *solidDB SmartFlow Data Replication Guide*. This guide describes how to use the Solid SmartFlow technology to synchronize data across multiple database servers.

- *solidDB AcceleratorLib User Guide*. Linking the client application directly to the server improves performance by eliminating network communication overhead. This guide describes how to use the AcceleratorLib library, a database engine library that can be linked directly to the client application.

This manual also explains how to use two proprietary Application Programming Interfaces (APIs). The first API is the Solid SA interface, a low-level C-language interface that allows you to perform simple single-table operations (such as inserting a row in a table) quickly. The second API is SSC API, which allows your C-language program can control the behavior of the embedded (linked) database server

This manual also explains how to set up a solidDB to run without a disk drive.

- *solidDB High Availability User Guide*. Solid CarrierGrade Option (formerly called the HotStandby Option) allows your system to maintain an identical copy of the database in a backup server or "secondary server". This secondary database server can continue working if the primary database server fails.

Chapter 2. Managing Data with solidDB

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

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

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

2.1 solidDB Data Management Components

solidDB includes the components described in the following sections.

2.1.1 Programming Interfaces (ODBC and JDBC)

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

ODBC

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

JDBC

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

Proprietary Interfaces

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

For more details on ODBC, JDBC, and Solid's propriety programming interfaces, read *solidDB Programmer Guide*.

2.1.2 Network Communications Layer

solidDB runs on all major network types and supports all of the main communication protocols (such as TCP/IP). Developers can create distributed applications for use in heterogeneous computing environments. Read Chapter 7, *Managing Network Connections*, for more details on network communication.

2.1.3 SQL Parser and Optimizer

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

Read Chapter 8, *Diagnostics and Troubleshooting* for more details on the Solid SQL Optimizer.

Optimizer Hints

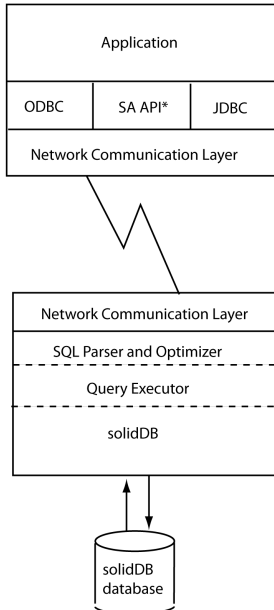
Optimizer hints (which are an extension of SQL) are directives specified through embedded pseudo comments within query statements. The optimizer detects these directives or hints and bases its query execution plan accordingly. Optimizer hints allow applications to be optimized under various conditions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

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

2.1.4 solidDB

The solidDB processes the data requests submitted via Solid SQL. The solidDB server shown in Figure 2.1, “solidDB Components” stores data and retrieves it from the database.

Figure 2.1. solidDB Components



* SA API is Solid's own API for use with the AcceleratorLib library.
For details, see solidDB AcceleratorLib User Guide.

2.1.5 System Tools and Utilities

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

Console Tools

Solid provides two ASCII-based console tools, Solid Remote Control (solcon) and Solid SQL Editor (solsql), to manage databases. These tools use a command line interface. Read Chapter 5, *Using Solid Data Management Tools* for details.

SolidConsole

SolidConsole is an easy-to-use graphical user interface for administering and monitoring Solid data management engines and executing SQL queries and commands. With SolidConsole, you can:

- execute SQL commands
- administer all database servers in a network from a single workstation
- generate backups either on-line or as a timed command
- obtain server status information
- use either the interactive or batch mode operation
- have multiple active connections to various servers
- save or print query results

In version 4.x of solidDB, the SolidConsole program is not provided as part of the package. Instead, you must download it separately from the Solid web site (<http://www.solidDB.com>).

Tools for Handling External ASCII data

solidDB provides the following tools for handling ASCII data:

- Solid SpeedLoader (solload) loads data from external ASCII files into a solidDB database. It is capable of inserting character data from character format. Solid SpeedLoader bypasses the SQL parser and uses direct writes to the database file with loading, which allows for fast loading speed.
- Solid Export (solexp) writes from a solidDB database to character format files. It is capable of writing control files used by Solid SpeedLoader to perform data load operations.
- Solid Data Dictionary (soldd) writes the data dictionary of a database. This tool produces a SQL script that contains data definition statements describing the structure of the database.

Read Chapter 5, *Using Solid Data Management Tools*, for details.

2.2 solidDB Architecture

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

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

2.2.1 Data Storage

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

Main Storage Tree

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

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

Solid Bonsai Tree Multiversioning and Concurrency Control

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

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

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

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

Index Compression

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

In-Memory Database Storage and In-Memory Indexes

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

2.2.2 Solid SQL Optimizer

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

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

Query Processing

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

Syntax Analysis

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

Creating the Execution Graph

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

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

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

Processing the Execution Graph

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

- Type-evaluation phase

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

- Estimate-evaluation phase

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

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

- Row-retrieval phase

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

2.2.3 Solid Network Services

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

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

Communication Session Layer

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

solidDB can listen to many protocols simultaneously.

2.2.4 Multithread Processing

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

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

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

Types of Threads

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

General Purpose Threads

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

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

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

Dedicated Threads

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

- I/O manager thread

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

- Communication read threads

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

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

There is usually one communication selector thread per protocol. Each running selector thread writes incoming requests into a common message queue.

- Communication server thread (also known as the RPC server main thread)

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

Chapter 3. Administering solidDB

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

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



Important

In the solidDB with AcceleratorLib, there are some differences in administration from standard solidDB. Wherever necessary, this chapter refers you to *solidDB AcceleratorLib User Guide* for AcceleratorLib-specific information.

3.1 What You Should Know

This section describes what you need to know about solidDB before you begin administration and maintenance.

3.1.1 Installing solidDB

If you have not yet installed solidDB, refer to the `releasenotes.txt` file delivered with the software. You find a detailed description of the installation in the `evaluation_setup.txt` file.

3.1.2 Using Solid Embedded Engine Databases 2.20 or Prior

Beginning with Solid Embedded Engine version 2.3 to the current version, the default collation sequence is set to the standard Latin-1. Solid Embedded Engine databases that were created with version 2.20 or prior do not match the Latin-1 collation sequence. To convert the data to Latin 1 in a version 2.20 database, you must export the database from its tables, extract data definitions, and load the tables to the new database. For details, read Section 5.8, “Tools Sample: Reloading a Database”.

3.1.3 Special Roles for Database Administration

solidDB has the following roles for administration and maintenance:

- `SYS_ADMIN_ROLE`

This is the Database Administrator role and has privileges to all tables, indexes, and users, as well as the right to use SolidConsole and Solid Remote Control (solcon). This is also the role of the creator of the database.

- `SYS_CONSOLE_ROLE`

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

- `SYS_SYNC_ADMIN_ROLE`

This is an administration role for performing administrative operations related to synchronization, such as deleting messages. ("Messages" are used to pass information back and forth between a master and its replicas. For example, to refresh the data that is in a master publication, the replica sends a REFRESH message.) Anyone with this access has all synchronization roles granted automatically. This role automatically includes the `SYS_SYNC_REGISTER_ROLE`.

- `SYS_SYNC_REGISTER_ROLE`

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

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

3.1.4 Automated and Manual Administration

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

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

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

Note that with SolidConsole's Administrative window, you can perform most of the SQL ADMIN COMMAND tasks that you execute on the command line with easy-to-use dialog boxes. For a description of SolidConsole, read Chapter 5, *Using Solid Data Management Tools*. Solid Remote Control (solcon) also lets you enter ad-

ministrative commands without using the ADMIN COMMAND syntax. See Section 5.2, “Solid Remote Control (solcon)” for details.

3.2 Starting solidDB



Note

This section applies to standard solidDB only. If you are using solidDB with AcceleratorLib, read the corresponding section in *solidDB AcceleratorLib User Guide*.

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

Table 3.1. Starting the Server

Operating System	To Start the Server...
UNIX Linux	Enter the command solid at the command prompt. When you start the server for the first time, enter the command solid -f at the command prompt to force the server to run in the foreground.
Open VMS	Enter the command run solid at the command prompt.
Microsoft Windows	Click the icon labeled solidDB in the solidDB program group.

For details on the *FileSpec* parameter, read the section called “FileSpec_[1...N] Parameter”.

3.3 Creating a New Database

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

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

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

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

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

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

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

Lowercase characters in the password are converted to uppercase. In other words, you can only have uppercase letters in the password.

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

See also Section 5.1, “Entering Password from a File”.



Note

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



Caution

The catalog name must not contain spaces.



Note

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

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

By default the database will be created as one file (`solid.db`) in the solidDB working directory. An empty database containing only the system tables and views uses approximately four megabytes of disk space. The time it takes to create the database depends on the hardware platform you are using. If you have a very small database (less than four megabytes) and want to keep the disk space less than four megabytes, set the value of the *ExtendIncrement* parameter in the `solid.ini` configuration file to less than 500 (default). This parameter and other parameters are discussed in Appendix A, *Server-Side Configuration Parameters*.

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

3.4 Login

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

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

3.5 About solidDB Databases

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

3.5.1 Solid Configuration File (`solid.ini`)

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

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

You can find a complete description of all parameters, details about the proper format of the `solid.ini` file, and instructions for specifying `solid.ini` configuration parameters in Appendix A, *Server-Side Configuration Parameters*. For more details about setting parameters, read Chapter 4, *Configuring solidDB*.

3.5.2 Setting Database BlockSize and Location

By default, solidDB databases set a block size for the database file as 8192 bytes (8KB). solidDB uses a multiple 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.

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

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

```
[Indexfile]
Blocksize=size_in_bytes
```

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

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

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

- location of the database file (the default is `solid.db` in the solidDB directory)
- maximum size (in bytes) the database file can reach (the default value is 2147483647, which equals 2GB - 1). The maximum file size is 4GB*blocksize - 1. With the default 8KB block size, this makes 32TB - 1.

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

For details on configuration with the *FileSpec* parameter, read Section 4.3.2, “Managing Database Files and Caching (*IndexFile* section)”.

3.5.3 Defining Database Objects

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

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

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

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

3.6 Connecting to solidDB



Note

This section applies to standard solidDB only. If you are using solidDB with AcceleratorLib, refer to the corresponding section in *solidDB AcceleratorLib User Guide*.

After starting solidDB, you can test the configuration by connecting to the server from your workstation using the Solid teletype tools, SQL Editor or Remote Control, or SolidConsole. Read Chapter 5, *Using Solid Data Management Tools*, for details on these utilities, which are part of the Solid Data Management tools.



Note

You must have SYS_ADMIN_ROLE or SYS_CONSOLE_ROLE privilege to be able to connect to a server using SolidConsole. For details on creating these roles, read the section in *solidDB SQL Guide* titled "Managing User Privileges And Roles".

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

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

Table 3.2. Connecting to solidDB

Tool	Command
Solid Remote Control (solcon)	<pre>solcon "networkname" [userid [password]]</pre> <p>For example:</p> <pre>solcon "tcp hobbes 1313"</pre> <p>If you did not specify the database administrator's user name and password on the command line, then solcon will prompt you to enter them.</p>
Solid SQL Editor (solsql)	<pre>solsql "networkname" [userid [password]]</pre> <p>For example:</p> <pre>solsql "tcp hobbes 1313"</pre> <p>If you did not specify the database administrator's user name and password on the command line, then solsql will prompt you to enter them.</p>
Solid Console	<pre>java solconsole -Ddatasenname -Url -uuserid -ppassword</pre> <p>For example:</p> <pre>java solconsole -Dsolid -Ujdbc:solid://localhost:1313 -udba</pre>

Tool	Command
	<p>-pdba</p> <p>Alternatively, you can start SolidConsole without any command line option. You are then prompted for the database connection information.</p> <p>In Windows you can start SolidConsole by clicking the StartSolidConsole labeled icon on the Desktop.</p>

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

3.7 Viewing the solidDB Message Log

Ensure the database started without errors by checking the message log `solmsg.out`, located in the Solid directory. You can view this file in SolidConsole's Messages page from the Administration window.

solidDB maintains the following message log files:

- The `solmsg.out` log file contains normal informational events, such as connects, disconnects, checkpoints, backups, failed logins etc. If an internal error occurs, the error is written to the `solmsg.out` file.
- If the error is fatal and causes the server to crash, then the `solerror.out` file contains more details about the error. Internal errors are selectively documented

3.7.1 Disabling Message Log Output

You can disable the generation of message log files. This is not advisable since it is difficult to diagnose problems without these files. Turning off message logging will increase performance and reduce disk space usage; however, in most cases the improvement is minimal. This option is useful only in unusual situations, such as when I/O is "expensive" (as it is in some systems that use FLASH memory), or in systems where data storage space is extremely limited and the message log file accumulates indefinitely without being deleted.

To disable log files, include the *DisableOutput* parameter in the *[Srv]* section of the `solid.ini` configuration file and set this parameter to yes. (By default, this parameter is set to "no".) If log file generation is already disabled, you can enable it by removing the parameter from the `solid.ini` file or setting the parameter to yes. The changes to the `solid.ini` file do not take effect until you restart the server.

3.7.2 Using Trace Files

For troubleshooting purposes, solidDB can also produce optional trace files that contain information for diagnostics. Monitoring the trace files is not necessary for everyday operation of the server. The trace files are primarily needed for troubleshooting of exceptional events. Refer to Chapter 8, *Diagnostics and Troubleshooting*, for more details on solidDB diagnostics.

3.7.3 Enabling Message Codes

Internally, 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]` parameter `PrintMsgCode` to "yes".

3.7.4 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 'FOO' tried to connect
```

from localhost.localdomain (127.0.0.1)
with an illegal username or password.



Note

The message code part is only included if message code printing is enabled in `solid.ini`. The SOLAPPINFO part is only included if the corresponding environment variable is set at the client computer.

3.8 Monitoring solidDB

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

3.8.1 Checking Overall Database Status

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

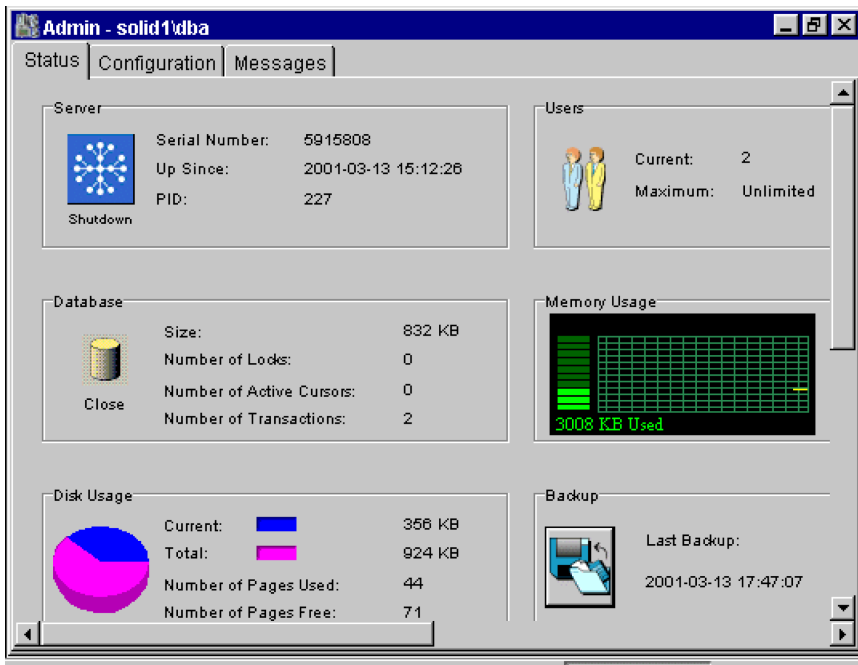
```
ADMIN COMMAND 'status';
RC TEXT
-- ----
0 SOLID solidDB started at Mon Feb 05 07:58:23 2001
0 Current directory is C:\work\java\commdemodb\clientDB
0 Using configuration file C:\work\java\commdemodb\clientDB\solid.ini
0 Memory statistics:
0     9778 kilobytes
0 Transaction count statistics:
0     Commit Abort Rollback      Total Read-only  Trxbuf  Active Validate
0       2426      0       475    2901      1876    382      1        0
0 Cache count statistics:
0     Hit rate      Find      Read      Write
0     100.0         167027    59       76
0 Database statistics:
0     Index writes      17377  After last merge    1218
0     Log writes        10771  After last cp        605
0     Active searches      7  Average              7
0     Database size      1232 kilobytes
0     Log size           1810 kilobytes
0 User count statistics:
0     Current Maximum Total
```

0 2 3 12

The result set fields are described below:

- Memory statistics show the amount of memory solidDB has allocated from the operating system. This number does not include the size of the executable itself.
- Transaction count statistics show the number of different transaction operations since startup.
- Cache count statistics show cache hit rate and number of cache operations since startup. Cache hit rate usually should be above 95 per cent. If it is below 95 per cent, consider increasing the cache size.
- Database statistics show a number of the most important database operations since startup. "Index writes after last merge" is an important figure here. It reveals the size of the multi-versioning storage tree of solidDB, known as the "Bonsai Tree." The smaller this value is, the better the server performance. A large value indicates that there is a long-running transaction active in the engine. Note that an excessively large Bonsai Tree causes performance degradation. For details on reducing Bonsai tree size, read Section 6.7, "Reducing Bonsai Tree Size by Committing Transactions".
- User count statistics shows the current and the maximum number of concurrent users.

You can select the Status option from the SolidConsole Administration window. The status page displays information as shown below.

Figure 3.1. SolidConsole: Status

3.8.2 Obtaining Currently Connected Users

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

ADMIN COMMAND 'userlist';

The command provides the following kind of result set:

RC TEXT

--

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

To obtain a list of currently connected users in SolidConsole:

1. Select the Status option from the SolidConsole Administration window or menu.
2. On the Status page, click the Users icon.

A Users dialog box displays each user's name, user id, type, machine id, and login time.



Note

If you are using the solidDB AcceleratorLib, "Linked" appears under Machine id.

3.8.3 Throwing Out a Connected solidDB User

To disconnect a single user from the server, perform one of the actions described below:

- Select the Status option from the SolidConsole Administration window or menu, click the Users icon, and drop a selected user from the Users dialog box.
- Enter the following command in SolidConsole or Solid SQL Editor (solsql):

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

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

3.8.4 Querying the Status of the Most Recent Backup

To obtain a status of the most recently run local backup, perform one of the following actions:

- Select the Status option from the SolidConsole Administration window, and click the Backup icon to view the backup status on the Backup dialog box.
- Enter the following command in SolidConsole or solsql :

```
ADMIN COMMAND 'status backup';
```

Obtaining a 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.

3.8.5 Detailed DBMS Monitoring (Perfmon)

One-time Monitoring Report

Besides checking the SolidConsole Status page, you can also take a snapshot that provides additional information on solidDB performance. Enter the following command in Solid SQL Editor:

ADMIN COMMAND 'perfmon';

The command returns a result set where each column represents a snapshot of the performance information that reflects the most recent few minutes. The command syntax also has options that allow you to specify output options. For details on these options, see the *perfmon* option syntax in Appendix E, *solidDB ADMIN COMMAND Syntax*.

The first column shows average performance information from a period of seconds. The "Total" column shows average information since solidDB was started. Most numbers are events/second. Those numbers that cannot be expressed as events/second (for example, database size) are expressed as absolute values.

There are more than one hundred counters and meters that can be studied. They can be categorized as follows:

- File operations
- Cache operations
- RPC and communications operations
- SQL operations
- SA (table-level db-operations) operations
- Transaction operations
- Index write (that is, database file write) operations
- Miscellaneous operations

You can restrict the output by providing a list of prefixes of counter names, like in:

```

admin command 'pmon db';
RC TEXT
-- ----
0 Performance statistics:
0 Time (sec)          43      43      42      30      30      44      42      33      Total
0 DBE insert      :    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
0 DBE delete      :    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
0 DBE update      :    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
0 DBE fetch       :    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.7
0 Db size         : 12032 12032 12032 12032 12032 12032 12032 12032 12032
0 Db free size    :  7816  7816  7816  7816  7816  7816  7816  7816  7816
8 rows fetched.

```

Producing a Continuous Performance Monitoring Report

One format of the **ADMIN COMMAND 'perfmon'** allows you to start and stop producing continuous performance counter reports to a file. The format is the following:

To start monitoring

```

ADMIN COMMAND 'perfmon
diff start filename interval [name_prefix_list]'

```

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

ADMIN COMMAND 'pmon diff start counter_log.csv 1000'

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

To stop monitoring:

ADMIN COMMAND 'pmon stop'

Full List of Perfmon Counters

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

Table 3.3. Perfmon Counters

Perfmon Variable	Description
Time (sec)	In one-time report: length of the measurement time interval, in seconds. The latest interval is on the right side of the table.
TimeMs	In a differential report: measurement time interval, in milliseconds. The oldest interval is in the first row of the table.
File open	File open calls/sec
File read	File read calls/sec
File write	File write calls/sec
File append	File append calls/sec
File flush	File flush calls/sec
File lock	File lock calls/sec
Cache find	Cache fetches/sec
Cache read	Cache misses/sec
Cache write	Cache page flushes/sec
Cache prefetch	Cache prefetched pages/sec
Cache prefetch wait	Cache waits for prefetched pages/sec
Cache preflush	Preflushing cache pages/sec
RPC messages	Total no. of sent messages/sec
RPC read	Total no. of read messages/s
RPC write	Total no. of write messages/sec
RPC uncompressed	When RPC compression enabled, no. of bytes/sec
RPC compressed	When RPC compression enabled, no. of compressed byte/s
Com sel empty	TCP socket select nil returns/sec
Com sel found	TCP socket select successes/sec
SQL prepare	SQL prepare statements/sec
SQL execute	SQL execute statements/sec
SQL fetch	SQL fetch statements/sec
DBE insert	Table engine row inserts/sec
DBE delete	Table engine row deletes /sec
DBE update	Table engine row updates /sec

Perfmon Variable	Description
DBE fetch	Table engine row fetches /sec
Proc exec	Procedure executions/sec
Trig exec	Trigger executions/sec
SA insert	SA-level row inserts/sec
SA delete	SA-level row deletes/sec
SA update	SA-level row updates/sec
SA fetch	SA-level row fetches/sec
Trans commit	Committed transactions/sec
Trans abort	Aborted transactions/sec
Trans rollback	Rolled back transactions/sec
Trans readonly	Read-only transactions/sec
Trans buf	Current transaction buffer size
Trans buf cleanup	Cumulative no. cleanup operations since startup
Trans buf added	Cumulative no. of transactions added since startup
Trans buf removed	Cumulative no. of transactions removed since startup
Trans validate	Current no. of active commit-time validations
Trans active	Current no. of active transactions
Ind write	Index writes/sec
Ind nomrg write	No. of nonmerged rows (committed and uncommitted)
Log write	Log record writes/sec
Log file write	Log block writes/sec
Log nopc write	Pending log records since last checkpoint
Log size	Total size of log file, in KB
Search active	Table engine-level active searches.
Db size	Total database size on disk, in KB
Db free size	Free space in the database (page level), in KB
Mem size	Total size of dynamically allocated memory, in KB
Merge quickstep	Quick merge steps/sec
Merge step	Full merge steps/sec

Perfmon Variable	Description
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 no. 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	Yes/no (1/0)
Checkpoint count	Checkpoint serial no. from startup
Checkpoint file write	Checkpoint file writes/sec
Checkpoint file read	Checkpoint file reads/sec
Est read samples	Estimator sample refresh call/s
Sync repl msg forw	Replica: get replies/s
Sync repl msg exec	Replica: execs/sec
Sync mast msg read	Master: message reads/sec
Sync mast msg exec	Master: message execs/sec
Sync mast msg write	Master: message writes/sec
Sync mast subs	Master: refreshes/sec
Log flush (L)	Logical log flushes/sec (e.g. commit)
Log flush (P)	Physical log flushes/sec
Log grpcommwkup	Group commit wakeups/sec
Log flush full	Log page full flushes/sec
Log wait flush	Current no. of user threads waiting for log operation
Log writeq full rec	Log writes while log write queue full (in no. of records)
Log writeq full byt (byte size)	Log writes while log write queue full (in bytes)

Perfmon Variable	Description
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 grouped acks	Secondary: current no. 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 no. 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: no. of log operations there is space for in the protocol window
HSB catchup reqcnt	HSB log write requests/sec, for catchup
HSB catchup waitcnt	HSB log waits-for-write requests/sec, for catchup
HSB catchup freespc	HSB: no. of log operations there is space for in the protocol window, for catchup
HSB alone freespc	Primary: in Primary alone, bytes there is room for in the transaction log
Thread count	Current no. of threads
Trans wait readlvl	Waits/sec for read level at commit
Lock ok	Successful lock requests/sec
Lock timeout	Lock timeouts/sec
Lock deadlock	Deadlocks/s
Lock wait	Lock waits/sec
MME cur num of locks	Current no. IME locks
MME max num of locks	Peak no. of IME locks (since startup)
MME cur num of lock chains	Current no. IME hash buckets
MME max num of lock chains	Peak no. IME hash buckets (since startup)
MME longest lock chain path	IME: longest hash overflow path
MME mem used by tuples	IME memory allocated to tuples
MME mem used by indexes	IME memory allocated to indexes

Perfmon Variable	Description
MME mem used by page structs	IME memory allocated to the shadow structures
B-tree node search keys	DBE B tree searches/sec
B-tree node search mismatch	DBE B tree searches/sec in mismatch index
B-tree node build mismatch	DBE B tree node rebuild/sec
B-tree node split	DBE B tree node splits/sec

3.8.6 Producing a Status Report

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

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

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

3.9 Shutting Down solidDB



Note

This section applies to standard solidDB only. If you are using the solidDB with AcceleratorLib, read the corresponding section in *solidDB AcceleratorLib User Guide*.

You can shut down the solidDB in the following ways:

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



Note

When using SolidConsole or Solid SQL Editor for steps 1-3 below, enter the full SQL Syntax,

```
ADMIN COMMAND 'command_name'
```

(for example, **ADMIN COMMAND 'close'**)

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

close

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

open

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

throwout all

Note that this command does not wait for open transactions to finish; it aborts and rolls back all open transactions.

3. Stop solidDB by entering the following command:

shutdown

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

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

3.10 Performing Backup and Recovery

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

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

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

3.10.1 Making Local Backups

You can initiate a local backup by entering the following command in the Query window of the SolidConsole or in the solsql:

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

Available options for the **backup** command:

Table 3.4. 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	<p><i>backup dir</i> is a path expression determining the backup directory in the local file system.</p> <p>If the backup directory is omitted, it must be specified in the <code>solid.ini</code> configuration file.</p> <p>If the specified backup directory does not exist, solidDB database error 10030 is given. For more information on this error, see Appendix D, <i>Error Codes</i></p>

The backup directory can be set beforehand in the configuration file by setting the parameter *BackupDirectory* in the *[General]* section of the configuration file. For the full list of available configuration parameters see Appendix A, *Server-Side Configuration Parameters*.

An additional way to make a backup is to select the Status option from the SolidConsole Administration window, and click the Backup icon to initiate the backup from the Backup dialog box.



Caution

If two databases are copied to the same directory, the earlier will be overwritten by the latter. The *backup dir* must be different at least for each database. Moreover, although database files may

be stored to different directories and partitions at the source server they all are copied to the same backup directory. Therefore equally named database files will conflict in the backup directory. As a consequence, only the last backed-up file among the equally named ones has backup copy in the backup directory.

3.10.2 Making Backups Over Network

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

Making NetBackup

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

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

Available options for the **netbackup** command:

Table 3.5. Options for the netbackup Command

Option	Description
-s	Synchronized execution. The call returns either when the netbackup is completed or due to an error.
connect	<i>connect str</i> is an elementary connect string specifying the connection to NetBackup Server. If the connect string is omitted it must be specified in the <code>solid.ini</code> configuration file.
dir	<i>backup dir</i> is a path expression determining the backup directory in NetBackup Server. The path can be either absolute or relative to the netbackup root directory. If the backup directory is omitted it must be specified in the <code>solid.ini</code> configuration file.
DELETE_LOGS	Delete backed-up log files in the source server. The backup using DELETE_LOGS is sometimes referred to as <i>Full backup</i> . This is the default value.

Option	Description
KEEP_LOGS	Keep backed-up log files in the source server. The backup using KEEP_LOGS is sometimes referred to as <i>Copy backup</i> . Using the keyword KEEP_LOGS corresponds to setting the <i>General</i> parameter <i>NetbackupDeleteLog</i> to "no".

For the full connect string syntax see the section called “Format of the Connect String”. For the full ADMIN COMMAND syntax see Appendix E, *solidDB ADMIN COMMAND Syntax*.



Caution

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

Flat or Deep NetBackup Directory Structure?

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

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

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

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



Tip

An easy way to retain the directory structure of the source server is to copy and rename the source server's `solid.ini` to `backup.ini` and move it to the backup directory at the NetBackup Server. The NetBackup Server reads only the `FileSpec_[1...N]` specifications from the [*IndexFile*]

section, creates similar directory structure and stores backup files with their original properties to the NetBackup Server.

3.10.3 Configuring and Automating Backups

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

Corresponding ADMIN COMMAND options and configuration parameters for local backup

Table 3.6. Parameter Correspondence to the `solid.ini` File for Local Backup

Option	Value	parameter in section <i>[General]</i> of <code>solid.ini</code>
dir	<code>backup dir</code>	BackupDirectory = <code>backup dir</code> default: no default

Corresponding ADMIN COMMAND options and configuration parameters for netbackup

Table 3.7. Parameter Correspondence to the `solid.ini` File for Netbackup

option	value	parameter in section <i>[General]</i> of <code>solid.ini</code>
connect	<code>connect str</code>	NetBackupConnect = <code>connect str</code> default: no default
dir	<code>backup dir</code>	NetBackupDirectory = <code>backup dir</code> default: no default
netbackup	DELETE_LOGS	NetbackupDeleteLog = yes default: yes
netbackup	KEEP_LOGS	NetbackupDeleteLog = no default: yes

For the complete list of configuration parameters and ADMIN COMMAND options see Appendix A, *Server-Side Configuration Parameters* and Appendix E, *solidDB ADMIN COMMAND Syntax*, respectively.



Note

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

Making backups can be automated by using timed commands. Read Section 3.14, “Entering Timed Commands” for details.

3.10.4 What Happens During Backup?

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

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

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

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



Note

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

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

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



Note

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

Local Backup

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

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



Caution

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

Network Backup

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

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

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

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

3.10.5 Administering Network Backup Server

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

```
NetBackupRootDir=netbackup root path
```

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

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

1. **ADMIN COMMAND 'close'**

No new netbackup requests are accepted.

2. **ADMIN COMMAND 'throwout all'**

Aborts the backups in progress.

3. **ADMIN COMMAND 'shutdown'**

Shuts down the server.

3.10.6 Monitoring and Controlling Backups

Solid offers a set of commands for monitoring and controlling backups. Backups can be controlled both by using the ADMIN COMMAND syntax in `solsql` and in Query window in Solid Console and by using the Administration window of the Solid Console.

Local Backup and Netbackup on Source-Server Side

You can query and control backup processes by using the ADMIN COMMAND -SQL extension in `solsql` or in the Query window of the Solid Console. The syntax is as follows:

```
ADMIN COMMAND 'command'
```

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

Table 3.8. Available Backup and Netbackup Commands

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

Example 3.1. Query the List of All Completed Backups and Their Success Status

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

ADMIN COMMAND 'backuplist'

Example 3.2. Abort an Active Network Backup Operation

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

ADMIN COMMAND 'abort netbackup'

You can control and monitor backup in Solid Console by selecting the Status sheet in the Administration window or menu and clicking the Backup icon. A backup status listing is displayed in a dialog box.

3.10.7 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". You can also query this information by using Solid Console.

If the backup failed, you can find the error message that describes the reason for the failure from the `solmsg.out` file in the database directory or in the Solid Console Messages page (accessed through the Administration window or menu). Correct the cause of the error and try again.

3.10.8 Typical Problems in Backups

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

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

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

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

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

3.10.9 Restoring Backups

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

Preparing NetBackup Files for Recovery

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

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

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

Returning to the State of the Last Backup

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

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

Refreshing Database from the Backup to the Current State

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

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

Recovering from Abnormal Shutdown

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

3.10.10 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 may be used when data loss from the last log-file disk block is affordable.

solidDB allows you to decide whether you want to use logging or not. If logging is used, abnormally shut down databases can be restored to the state they were at the moment the failure took place. If the logging is disabled, databases can be restored to the backup state only. Transaction logging is enabled by default. If the

full transaction recovery is not needed, logging can be disabled. To do this, set the *[Logging]* parameter *LogEnabled* to "no".

Logging may be synchronous or asynchronous, depending on the transaction durability setting. For more on transaction durability, see the subsection Logging and Transaction Durability in Chapter 6, *Performance Tuning*.

3.11 Creating Checkpoints

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

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

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

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

ADMIN COMMAND 'makecp'

(Make CheckPoint). You can also force a checkpoint using a timed command. Read Section 3.14, "Entering Timed Commands" for details.

solidDB has an automatic checkpoint creation daemon, which creates a checkpoint after a certain number of writes to the log files. For more information about controlling the frequency of checkpoints, see Section 6.6, "Tuning Checkpoints".

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



Note

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

A checkpoint can require a substantial amount of I/O, and may affect the server's responsiveness while the checkpoint is occurring. For more details, read Section 6.6, “Tuning Checkpoints”.

3.12 Closing a Database

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

ADMIN COMMAND 'close';

You use the close command when you want to prevent users from connecting to the database. For example, when you are shutting down solidDB, you must prevent new users from connecting to the database. As part of the shut down procedure you use the close command. Read Section 3.9, “Shutting Down solidDB” for procedures to shut down a database.

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

To revert the effect of the **close** command, use:

ADMIN COMMAND 'open';

SolidConsole provides a user interface for managing database connections. For details, Refer to SolidConsole Online Help available by selecting Help on the menu bar.

3.13 Running Several Servers on One Computer

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

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

3.14 Entering Timed Commands

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

3.14.1 To Enter a Timed Command Manually

Edit the *At* parameter of the *[Srv]* section in the *solid.ini* file. The syntax is:

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

If the day is not given, the command is executed daily. For details on valid commands, refer to the table at the end of this section.

Example:

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



Note

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

3.14.2 To Enter a Timed Command in SolidConsole

Select the Status option from the SolidConsole Administration window or menu, click the Scheduler icon, then click New to enter a timed command in the Scheduler dialog box.

In the Scheduler dialog box, provide the day, time, command and arguments in each of the applicable fields. For syntax details, refer to the previous section. Refer to the following table for a list of valid commands.

Table 3.9. 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	system command	no default
open	no argument	no default
close	no argument	no default

3.15 Compacting the Database Files

3.15.1 What Is Database Reorganization?

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

3.15.2 How Does the Database Reorganization Work?

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

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

3.15.3 Database Reorganization Command Line Options

Free Factor Report

```
solid -x infodbfreefactor
```

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

Reorganization

```
solid -x reorganize
```

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

See Appendix C, *solidDB Command Line Options* for other utilities invoked with a command line option.

3.16 Encrypting a Database

A symmetric key data encryption method can be used to encrypt the database pages. This feature can be used to protect sensitive data against a device theft. The product is shipped with a weak DES (single DES) algorithm because of export restrictions applying. This algorithm is not recommended to be used for demanding applications requiring strong security.

3.16.1 Enabling Encryption

Encryption of the entire database can be enabled when the server is started using command line options `-E` and `-S`. The `-E` option invokes database encryption if the database used is not encrypted. The `-S` option protects the symmetric encryption key.

3.16.2 Protecting the Encryption Key

The symmetric encryption key is stored in the unencrypted header page of the database file. To protect the symmetric encryption key, a startup password must be specified with either the `-S` option or with **`-x startp-wdfile`**. The startup password is mandatory whenever `-E` is specified. If the password is given, the minimum length is three characters. There is an option to specify an empty password whereby the encryption key is left unprotected.

3.16.3 Creating an Encrypted Database

You can create an encrypted database by using options `-E` and `-S` as follows:

```
solid -E -S <startup password>
```

A safer way is to use options `-E` and `-x keypwdfile:<filename>`

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

3.16.4 Starting an Encrypted Database

To start an already existing encrypted database, the `-S` option must be used. Otherwise the server prompts the user for the startup password.

The startup password is specified with a command line option as follows:

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

or using a file and option:

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



Note

Use the `-x keypwdfile` option instead of option `-S`. Using the password in the command line is not secure on most of the systems. For example in UNIX systems, other users might see the password in the `ps` command output. Use command line option `-S` for debugging or evaluation purposes only.

3.16.5 Changing the Encryption Key Password

To change the password of the encryption key, the server must be started using option `-E` and the old and the new password must be given using option `-S` as follows:

```
solid -E -S <old password> -S <new password>
```

An alternative and recommended way to change the startup password is to specify a password file twice with **-x keypwdfile**:

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



Note

To turn off encryption key protection, the password can be replaced with an empty password.

3.16.6 Decrypting a Database

It is possible to decrypt the database with option **-x decrypt**. A startup password is mandatory for database file decryption:

```
solid -x decrypt -S <password>
```

or

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

3.16.7 Encryption Query

Some application systems do not allow storing data in an unencrypted file. The application can check the security level of the database data before, for example, registering a new replica. For this purpose, there is function

```
database_encryption_level()
```

that has the following return values:

0 - no encryption

1 - encrypted, the key is not protected (empty password)

2 - encrypted, the key is protected by a separate startup password

3 - encrypted, a custom encryption method is used (for accelerator only)

3.16.8 Backups

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

3.16.9 HSB

HSB traffic is not encrypted by means of database file encryption. To protect the HSB traffic, other security means are needed.

When making an HSB copy or netcopy, the database file and logs are transferred in encrypted form to avoid redundant encryption/decryption of the files. Theoretically, it is possible to have an HSB server pair having different encryption keys (and even different algorithms), but that is not desirable. The recommended procedure is to encrypt the Primary database first and then copy or netcopy it.

3.16.10 Accelerator

The Accelerator API is extended with an interface for setting custom encryption algorithms. Function SSCSetCipher sets application-provided encryption and decryption functions for the accelerator. It must be invoked before the server is started with SSCStartServer.

```
void SSC_CALL SSCSetCipher(  
    void* cipher,  
    char* (SSC_CALL *encrypt)(void *cipher, int page_no, char *page,  
        int n, size_t pagesize),  
    int (SSC_CALL *decrypt)(void *cipher, int page_no, char *page,  
        int n, size_t pagesize));
```

cipher - cipher refers to the application provided security context (cipher object), such as the encryption password. The same parameter is passed back to the application-provided encryption/decryption functions.

encrypt - encryption function. Returns its page parameter.

decrypt - decryption function. Returns a non-zero value or the server exits with "password mismatch" error.

page_no - number of the page being encrypted/ decrypted. The application is likely to ignore this parameter or it might be used as an additional encryption/decryption parameter.

page - pointer to the area to be encrypted/decrypted by the application functions.

n - number of the pages to be encrypted/decrypted

pagesize - size of the page to be encrypted/decrypted

3.16.11 Performance Impact

Using an encrypted database affects the database server performance for both read and write operations. Performance impact on read-operations is mostly determined by the cache hit rate and is not significant when the cache hit rate is good.

For insert and update operations, the server always has to encrypt and decrypt the log files (if they are used) and in this case performance penalty can be more significant.

Chapter 4. Configuring solidDB

This chapter describes how to configure solidDB to meet your environment, performance, and operation needs. It includes the most important parameters and their settings. See Section 4.4, “Managing Server-Side Parameters” for step-by-step instructions on how to view and set the parameter values by using Solid Remote Control (solcon), SQL Editor (solsql), or SolidConsole.



Important

If you are using the solidDB with AcceleratorLib, please refer to solidDB AcceleratorLib User Guide for more information on parameters that are specific to AcceleratorLib.

If you are using the solidDB with the CarrierGrade (HotStandby) option, please refer to *solidDB High Availability User Guide* for information on parameters that are specific to the CarrierGrade option.

4.1 Configuration Files and Parameter Settings

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



Note

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

When solidDB starts, it attempts to open `solid.ini` first from the directory set by the `SOLIDDIR` environment variable. If the file is not found from the path specified by this variable or if the variable is not set, the server or client attempts to open the file from the current working directory. (The current working directory is normally the same as the directory from which you started the solidDB server, or a client application. You may specify a different working directory by using the “-c” server command-line option. For more information about command-line options, see *Appendix B, solidDB Command Line Options* in *solidDB Administration Guide*.)

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

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

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

```
Listen=tcp 192.168.255.1 1315
```

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

Parameters are grouped according to section categories in the configuration file. See *Appendix A, Server-Side Configuration Parameters* and *Appendix B, Client-Side Configuration Parameters* in *solidDB Administration Guide*. for an overview of the section categories and all available parameters

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

```
[com]
```

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

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

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

4.2 Most Important Client-Side Parameters

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

4.2.1 Defining Network Names (Com section)

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

Connect Parameter

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

The default value is Operating System dependent. Refer to Chapter 7, *Managing Network Connections*.

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 Solid ODBC Driver, the ODBC Data Source Name is used and the *Connect* parameter has no effect.

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

Format of the Connect String

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

Connect string format:

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

where options can be any number of:

Table 4.1. Connect String Options

Option	Meaning
-z	Data compression is enabled for this connection
-c <i>milliseconds</i>	Login timeout is specified (the default is operating-system-specific). A login request fails after the specified time has elapsed. Note: applies for the tcp protocol only.

Option	Meaning
<code>-r milliseconds</code>	Connection (or read) timeout is specified (the default is 60 s). A network request fails when no response is received during the time specified. The value 0 sets the timeout to infinite. Note: applies for the tcp protocol only.

Examples:

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

Trace Parameter

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

TraceFile Parameter

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

4.3 Most Important Server-Side Parameters

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

4.3.1 Defining Network Names (Com section)

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

Listen parameter

The *Listen* parameter in the *[Com]* section defines the network name for the server; this is the protocol and name that a solidDB server uses when it starts to listen to the network. Client processes communicate

with the server using this network name. The default value is Operating System dependent. Refer to Chapter 7, *Managing Network Connections*, for details on the parameter format.

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

4.3.2 Managing Database Files and Caching (*IndexFile* section)

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

FileSpec_[1...N] Parameter

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

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

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

The default value for this parameter is

```
solid.db 2147483647
```

(which equals 2 GB-1 expressed in bytes)

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

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

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

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



Note

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

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

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

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

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

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

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

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

```
FileSpec_1=C:\solddb\solid.1 1000M 1
```



```
FileSpec_2=D:\solddb\solid.2 1000M 2  
FileSpec_3=E:\solddb\solid.3 1000M 2
```

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

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



Caution

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

CacheSize

The *CacheSize* parameter defines the amount of main memory the server allocates for the cache. The default value depends on the server operating system. The minimum size is 512 kilobytes. For example:

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

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

The default cache size is 32 MB.

4.3.3 Specifying the Local Backup Directory (General Section)

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

BackupDirectory Parameter

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

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

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



Note

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

4.3.4 Specifying the Network Backup Directory (General section)

These parameters set the target directory in the NetBackup Server for the backup files, log files and the configuration file. If the remote directory doesn't exist, it is created if possible.

Source-Side Parameter

The parameter

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

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

NetBackup Server-Side Parameter

The parameter

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

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



Important

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

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

FileNameTemplate

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

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

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



Note

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

The filename can also be structured by using the *FileNameTemplate* parameter together with the *LogDir* parameter, in which case the *LogDir* parameter defines the directory prefix of the filename and the *FileNameTemplate* parameter defines the actual filename. For more information, see Section A.10, “Logging Section”.

4.3.5 Specifying a Directory for the External Sorter Algorithm (Sorter Section)

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

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

TmpDir_[1...N]

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

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

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

4.3.6 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 Section 2.2.4, "Multithread Processing" for more details.

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

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

threads= (2 x number of processors) + 1

Threads

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

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

4.3.7 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 *solidDB SQL Guide*.

Info

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

4.3.8 Specifying Network Communication Tracing (Com Section)

The communication tracing facility is necessary, for instance, if the network hardware is not functioning properly. By turning the tracing on, the communication layer is capable of logging even the system specific errors and may help in diagnosing the real problem in the network. For details, read Section 8.1.1, “The Network Trace Facility”. The following parameters control the outputting of network trace information.

Trace

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

TraceFile

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

4.4 Managing Server-Side Parameters

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

- Entering the commands:

ADMIN COMMAND 'parameter'

and

ADMIN COMMAND 'describe parameter'

in SolidConsole or Solid SQL Editor (teletype).

- Using the Administration→Configuration page in SolidConsole.

The SolidConsole Configuration page lets you display a parameters listing in a tree node format and change configuration settings through a dialog box. For details, refer to SolidConsole Online Help available by selecting Help on the menu bar.

- Directly, by editing the `solid.ini` file in the Solid directory.

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



Note

For details on viewing and setting server communication protocol parameters only, read Chapter 6, "Managing Network Connections" in *solidDB Administration Guide*.

4.4.1 Viewing and Setting Parameters with ADMIN COMMAND

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

Viewing Parameters

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

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

where:

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

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

ADMIN COMMAND 'parameter';

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

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

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

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

The three values shown are (in this order):

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

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

ADMIN COMMAND 'parameter -r';

Viewing the Description of a Specific Parameter

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

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

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

A result set for a single parameter looks like this:

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

The rows of the resultset are:

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

- *Factory value* displays the value preset in the product.

Setting a Parameter Value

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

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

where:

value is a valid parameter value.



Note

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

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

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

Example:

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



Note

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

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

Persistence of Parameter Modifications

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

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

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

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

4.4.2 Viewing and Setting Parameters in `solid.ini`

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

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

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

You must restart the server to activate the changes.

4.4.3 Constant Parameter Values

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

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

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

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

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



Note

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

Chapter 5. Using Solid Data Management Tools

This chapter describes Solid Data Management Tools, a set of utilities for performing various database tasks. These tools include:

- Solid Remote Control (solcon) and Solid SQL Editor (solsql) for command line sessions at the operating system prompt.
- SolidConsole, an easy-to-use graphical user interface for administration and configuration tasks, monitoring local and remote solidDB servers, issuing SQL queries and statements, and executing SQL script files.
- Solid SpeedLoader (solload) for loading data from external ASCII files into a solidDB database.
- Solid Export (solexp) for unloading data from a solidDB database to ASCII files.
- Solid Data Dictionary (soldd) for retrieving data definition statements from a solidDB database.



Note

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



Note

Not all Solid Tools are necessarily part of the standard product delivery, and their availability on some platforms may be limited. For information about Solid data management tools, contact your Solid sales representative or Solid Online Services at the Solid Web site:

<http://www.solidDB.com>

5.1 Entering Password from a File

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

The syntax is as follows:

command -x pwdfile:filename

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

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

Command examples:

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

5.2 Solid Remote Control (solcon)

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



Note

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

5.2.1 Starting Solid Remote Control

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

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

solcon options servername username password

where *options* can be:

Table 5.1. solcon Command Options

Option Syntax	Description
-cdir	Change working directory.
-ecommand string	Execute the specified Remote Control command.
-ffilename	Execute command string from a script file.
-x pwdfile:filename	Read password from the filename.
-h, -?	Help = Usage.

Servename is the network name of a solidDB server that you are connected to. Logical Data Source Names can also be used with tools; refer to Chapter 7, *Managing Network Connections* for further information. The given network name must be enclosed in quotes.

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

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

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

To exit Remote Control, enter the command **exit**.

Example 5.1. Remote Control

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

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

Start up Remote Control to back up a specific database:

```
solcon -ebackup 'ShMem SOLID' dbadmin password
```

5.2.2 Entering Commands in Solid Remote Control

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

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

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

```
backup
```

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

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

Table 5.2. Remote Control Specific Commands

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

5.3 Solid SQL Editor (solsql)

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

ADMIN COMMAND 'help'

5.3.1 Starting Solid SQL Editor

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

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

```
solsql options servername username password
```

where options can be:

Table 5.3. solsql Command Options

Option Syntax	Description
-a	Auto commit every statement.
-cdir	Change working directory.
-esql-string	Execute the SQL string; if used commit can only be done using -a.
-ffilename	Execute SQL string from a script file.
-h, -?	Help = Usage.
-ofilename	Write result set to this file.
-Ofilename	Append result set to this file.
-sschema_name	Use only this schema.
-t	Print execution time per command.
-u	Expect input in UTF-8 format.
-x pwdfile:filename	Read password from the filename.
-x onlyresults	Print only rows.

**Note**

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

Servename is the network name of a solidDB server that you are connected to. Logical Data Source Names can also be used with tools; Refer to Chapter 7, *Managing Network Connections* for further information. The given network name must be enclosed in double quotes.

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

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

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

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

To exit SQL Editor, enter the command **exit**.

Running SQL Scripts

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

@filename

For example:

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

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

Example 5.2. SQL Script Examples

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

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

Start SQL Editor and execute the tables.sql script:

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

5.3.2 Executing SQL Statements with Solid SQL Editor

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

Example:

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



```
commit work;

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

drop table testtable;
commit work;
```

Executing a SQL Script from a File

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

```
solsql servername username password filename
```

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

Example:

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



Note

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

5.4 SolidConsole

SolidConsole is a java-based, graphical user interface for managing, administering, and querying local and remote solidDB servers. Designed for intuitive and efficient ease-of-use, it allows you to create and manipulate database schemas, browse data, monitor and manage both local and remote databases, and configure solidDB server parameters.

With SolidConsole you can use an Administration window, which features an intuitive interface to perform the basic administration tasks described in this manual. You can also use a Query window to issue Solid SQL ADMIN COMMANDs for task administration and enter SQL statements and queries to create and execute script files.

**Note**

Performing administration operations in SolidConsole requires SYS_ADMIN_ROLE rights.

5.4.1 Starting SolidConsole

To start SolidConsole enter the following command at your operating system prompt:

```
java -classpath .\solconsole.jar;.\SolidDriver2.0.jar;. solconsole
```

or when using Microsoft Windows, start SolidConsole from the icon in your Program Group. In Linux, it can also be launched by entering

```
run.sh
```

in the SolidConsole working directory.

You can launch the SolidConsole by including one or more of optional command line arguments:

```
java [javaoptions] solconsole [options]
```

where options can be:

Table 5.4. solconsole Command Options

Option Syntax	Description
-Mmode	mode = BATCH; specifies that SolidConsole run in Batch Mode without showing the user interface.
-Ddatabasename	Specifies a database for connection.
-Uurl	<p>Specifies the JDBC URL required for SolidConsole to connect to a solidDB server. The format of the JDBC URL is:</p> <p><i>JDBC:SOLID://machine_name:port_number</i></p> <p>For example:</p> <p><i>jdbc:solid://localhost:1313</i></p>

Option Syntax	Description
-userid	Specifies the user ID for accessing the database
-ppassword	Specifies the user's password for accessing the database
-fqueryfile	Executes the SQL statements contained in the script file.
-esql_strings	Executes the SQL strings.
-oOutputfile	Specifies the file where resultsets are stored.
-OOutputfile	Specifies the file (which will be opened for writing in append mode) where resultsets are stored.
-a	All transactions are autocommitted.
-h	Help Usage.



Note

Ensure that the solidDB server is running before establishing a database connection. Use the Add Database dialog box to add additional databases and the Connect dialog box to connect to the databases. For details, refer to SolidConsole Online Help available by selecting Help on the menu bar.

5.4.2 SolidConsole Interface Features

SolidConsole opens each new database connection with three separate windows: a Browse window, a Query window, and an Administration window. You can move from one window to another to manage different databases simultaneously.



Note

The features of each window are described briefly in the following sections. For details on usage, refer to the SolidConsole Online Help available by selecting Help on the menu bar.

Query Window

With the Query window, you can issue solidDB SQL ADMIN COMMANDs to perform administration tasks, issue SQL queries and statements, or execute a script file that contains queries and statements. For a list of administrative commands you can use in SolidConsole, refer to the ADMIN COMMAND section in solidDB SQL Guide.

A results section in the Query window displays error messages and the result set, which you can print or save to a text file. If needed, you can cancel execution of a current SQL statement and specify transaction commits

and rollbacks. Settings are also available to enable autocommit and the transaction isolation level for a connection.

Administration Window

With the Administration window, you can monitor server status (including messages) and control all solidDB servers in a network from a single workstation. From the Administration window, you can perform the following local and remote operations:

- Control user access to databases
- Control network protocol connections
- Generate backups and checkpoints
- Create timed commands to automate administration
- Configure a solidDB server's parameters

Browse Window

With the Browse window, you can browse database objects, which include tables, columns, views, indexes, stored procedures, sequences, roles, and users. A database workspace gives you a quick view of database connections, databases, and their objects in a tree format. You can click on a node in the tree to browse an object, which is displayed in table format. For easier viewing, you can rearrange data columns by moving and resizing table headers.

5.5 Solid SpeedLoader

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

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

Please note the following:

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

catalog_name.schema_name.table_name

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

However, Solid SpeedLoader always checks for unique or foreign key constraints that are defined using the CREATE TABLE statement. For more details on constraints, see the CREATE TABLE syntax in the "solid SQL Syntax" appendix in solidDB SQL Guide.

5.5.1 Control File

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

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



Note

Each import file requires a separate control file. Solid SpeedLoader loads data into one table at a time.

For more details about the control file format, read the section called “Control File Syntax”.

5.5.2 Import File

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

- In fixed-length format data records have a fixed length, and the data fields inside the records have a fixed position and length.

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

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

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

5.5.3 Message Log File

During loading, Solid SpeedLoader produces a log file containing the following information:

- Date and time of the loading
- Loading statistics such as the number of rows successfully loaded, the number of failed rows, and the load time if it has been specified with the option
- Any possible error messages. For details on SpeedLoader errors, see Section D.14, “Solid SpeedLoader Utility (solload) Errors”.

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`.

5.5.4 Configuration File

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

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

```
[Com]  
Connect=netbios SOLID
```

5.5.5 Starting Solid SpeedLoader

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

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

where options can be:

Table 5.5. solload Command Options

Option Syntax	Description
-brecords	Number of records to commit in one batch
-cdir	Change working directory
-Ccatalog_name	Set the default catalog from where data is read from or written to.
-lfilename	Write log entries to this file.
-Lfilename	Append log entries to this file.
-nrecords	Insert array size (network version).
-sschema_name	Set the default schema.
-t	Print load time.
-h	Help = Usage.
-x emptytable	Load data only if there are no rows in the table.
-x errors:count	Maximum error count.
-x nointegrity	No integrity checks during load.
-x pwdfile:filename	Read password from the file.
-x skip:records	Number of records to skip.
-xutf8	WCHAR data is in UTF-8 format.

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

Servename is the network name of a solidDB server that you are connected to. Logical Data Source Names can also be used with tools; Refer to Chapter 7, *Managing Network Connections* for further information. The given network name must be enclosed in quotes.

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

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

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

Control File Syntax

The control file syntax has the following characteristics:

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

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

Table 5.6. SpeedLoader Reserved Words

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

TINYINT	VARBIN	VARCHAR	WHITESPACE
---------	--------	---------	------------

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

Table 5.7. Full Syntax of the Control File

Syntax Element	Definition
<i>control_file</i>	<pre> ::= [option_part] load_data_part into_table_part fields column_list </pre>
<i>option_part</i>	<pre> ::= OPTIONS (options) </pre>
<i>options</i>	<pre> ::= option [, option] </pre>
<i>option</i>	<pre> ::= [SKIP = int_literal] [ERRORS = int_literal] </pre>
<i>load_data_part</i>	<pre> ::= LOAD [DATA] [characterset_specification] [DATE date_mask] [TIME time_mask] [TIMESTAMP timestamp_mask] [INFILE filename] [PRESERVE BLANKS] </pre>
<i>characterset_specification</i>	<pre> ::= CHARACTERSET { NOCONVERT </pre>

Syntax Element	Definition
	<pre> NOCNV ANSI MSWINDOWS PCOEM IBMPCL SCAND7BIT } </pre> <p>Note that UTF8 is not allowed inside the control file.</p>
<i>into_table_part</i>	<code>::= INTO TABLE <i>tablename</i></code>
<i>fields</i>	<code>::= [FIELDS {<i>termination</i> <i>enclosure</i>}]</code>
<i>termination</i>	<code>::= TERMINATED BY <i>termination_char</i> [[OPTIONALLY] <i>enclosure</i>]</code>
<i>termination_char</i>	<code>::= WHITESPACE 'char' "char" <i>hex_literal</i></code>
<i>enclosure</i>	<code>::= ENCLOSED BY <i>enclose_char</i> [AND <i>enclose_char</i>]</code>
<i>enclose_char</i>	<code>::= 'char' "char" <i>hex_literal</i></code>
<i>hex_literal</i>	<code>::= X'hex_byte_string'</code>

Syntax Element	Definition
<i>column_list</i>	<code>::= column [, column]</code>
<i>column</i>	<code>::= column_name datatype_spec</code> <code>[POSITION (int_literal { : - } int_literal)]</code> <code>[DATE date_mask]</code> <code>[TIME time_mask]</code> <code>[TIMESTAMP timestamp_mask]</code> <code>[NULLIF BLANKS NULLIF NULLSTR NULLIF 'string' </code> <code>NULLIF ((int_literal { : - } int_literal) = 'string')]</code>
<i>datatype_spec</i>	<code>::= { BINARY CHAR [(length)] DATE </code> <code>DECIMAL [(precision [, scale])] </code> <code>DOUBLE PRECISION FLOAT [(precision)] INTEGER </code> <code>LONG VARBINARY LONG VARCHAR </code> <code>NUMERIC [(precision [, scale])] </code> <code>REAL SMALLINT TIME </code> <code>TIMESTAMP [(timestamp precisionv)] </code> <code>TINYINT VARBINARY VARCHAR [(length)] }</code>

The following paragraphs explain syntax elements and their use in detail.

CHARACTERSET

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



Note

UTF8 is not allowed inside the control file.

DATE, TIME, and TIMESTAMP

These keywords can be used in two places with different functionality:

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



Note

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

Table 5.8. Data Masks

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

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

Example 5.3. Date Example in Control File

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

```
OPTIONS ( SKIP=1 )
```

```
LOAD DATA  
RECLLEN 12
```

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

Example 5.4. Date, Time, and Timestamp Examples in Control File

Note that the following example uses the `FIELDS TERMINATED BY` keyword. For details on this keyword, read the section called “`FIELDS TERMINATED BY`”.

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

PRESERVE BLANKS

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

INTO_TABLE_PART

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

FIELDS ENCLOSED BY

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

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

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

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

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

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

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

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

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

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

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

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

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

the input must be:

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

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

X 'hex_byte_string'

For example:

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

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

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

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

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

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

```
"Hello."  
Hello.
```

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

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

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

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

ENCLOSED BY Input Rules and Examples

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

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

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

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

Example 5.5. Treatment of Enclosed Characters within the Data

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

If the input file contains:

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

it produces the following format in the database:

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

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

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

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

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

Example 5.6. Treatment of Final Enclosing Character

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

To get the following format in the database:

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

the input file must contain:

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

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

Example 5.7. Embedding Newline Characters

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

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

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

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

FIELDS TERMINATED BY

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

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

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

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

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

or

```
1:2:3
```

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

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

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

In the example above, the separator is a comma.

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

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

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

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

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

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

Data File Example

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

Control File Example

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

POSITION

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

NULLIF

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

Example 5.8. 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
)
```

Example 5.9. 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.

5.5.6 Loading Fixed-Format Records

Example 5.10. Control File Example 1

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

OPTIONS (ARRAYSIZE=3)

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

```
        ID        POSITION(11-15)
)
```

Example 5.11. Control File Example 2

```
-- EXAMPLE 2
OPTIONS (SKIP = 10, ERRORS = 5)
-- Skip the first ten records. Stop if
-- errorcount reaches five.
LOAD DATA
INFILE 'sample.dat'
-- import file is named sample.dat
INTO TABLE TEST1 (
  ID        INTEGER 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)
```

5.5.7 Loading Variable-Length Records

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

Example 5.12. Control File Example 3

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

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

```
        ID
    )
```

Example 5.13. Control File Example 4

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

```
-- If the column for DATE2 is 'NULL' a NULL is  
-- inserted.
```

5.5.8 Running a Sample Load Using Solload

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

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

```
solload 'shmem solid' dba dba delim.ctr
```

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

```
solload 'shmem solid' dba dba fixed.ctr
```

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

```
Solid Speed Loader v.4.10.00xx  
(C) Copyright Solid Information Technology Ltd 1992-2003  
Load completed successfully, 19 rows loaded.
```

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

```
Solid Speed Loader v.4.10.00xx  
(C) Copyright Solid Information Technology Ltd 1992-2003  
Load completed successfully, 19 rows loaded.
```

5.5.9 Hints to Speed up Loading

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

- Connect locally if possible; it is faster not to load data over the network.

- Increase the number of records committed in one batch. By default, commit is done after each record.
- Disable transaction logging.

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

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

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

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



Note

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

5.6 Solid Export

Solid Export is a product for unloading data from a solidDB database to ASCII files. Solid Export produces both the import file, that is, the file containing the exported ASCII data, and the control file that specifies the format of the import file. Solid SpeedLoader can directly use these files to load data into a solidDB database.



Note

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

5.6.1 Starting Solid Export

Start Solid Export with the command `solexp`. If you start Solid Export with no arguments, you'll see a summary of the arguments with a brief description. The command line syntax is:


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

where options argument can be:

Table 5.9. solexp Command Options

Option Syntax	Description
-cdir	Change working directory
-esql_string	Execute SQL string for export.
-ffilename	Execute SQL string from file for export.
-lfilename	Write log entries to this file.
-Lfilename	Append log entries to this file.
-ofilename	Write exported data to this file.
-sschema_name	Use only this schema for export.
-Ccatalog_name	Set the default catalog from where data is read from or written to.
-p	Preserve case of schema and table names.
-8	Output 8-bit names to .crt file (disables UNICODE names).
-h, -?	Help = Usage.
-x pwdfile:filename	Read password from the file.



Note

1. The symbol * can be used to export all tables with one command. However, it cannot be used as a wildcard.
2. The **-ttablename** (Export table) option is still supported in order to keep old scripts valid.

Servername is the network name of a solidDB that you are connected to. Logical Data Source Names can also be used with tools; refer to Chapter 7, *Managing Network Connections* further information. The given network name must be enclosed in double quotes.

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

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

For example:

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

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

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.

5.7 Solid Data Dictionary

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



Note

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

5.7.1 Starting Solid Data Dictionary

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

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

where options can be:

Table 5.10. soldd Command Options

Option Syntax	Description
-cdir	Change working directory
-ofilename	Write data definitions to this file.

Option Syntax	Description
-Ofilename	Append data definitions to this file.
-Ccatalog_name	Set the default catalog from where data definitions are read from or written to.
-sschema_name	List definitions from this schema only.
-p	Preserve case of schema and table names.
-8	Output 8-bit names to .crt file (disables UNICODE names).
-h, -?	Help = Usage.
-x tableonly	List table definitions only.
-x indexonly	List index definitions only.
-x viewonly	List view definitions only.
-x sequenceonly	List sequence definitions only.
-x procedureonly	List procedure definitions only.
-x publicationonly	List publication definitions only.
-x eventonly	List event definitions only.
-x triggeronly	List trigger definitions only.
-x schemaonly	List schema definitions only.
-x hiddennames	List internal constraint names only.
-x pwdfile:filename	Read password from the file.

Servename is the network name of a solidDB server that you are connected to. Logical Data Source Names can also be used with tools; Refer to Chapter 7, *Managing Network Connections* for further information. The given network name must be enclosed in quotes.

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

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

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

Example 5.14. Solid Data Dictionary Examples

```
soldd -odatabase.sql "tcp database_server 1313" dbadmin flq32j4
```

Print the definition of procedure TEST_PROC:

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



Note

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

5.8 Tools Sample: Reloading a Database

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

5.8.1 To Reload the Database

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

Example 5.15. Reload the Database: Walkthrough

In this example the server name is Solid and the protocol used for connections is Shared Memory. Therefore, the network name is 'shMem SOLID'. The database has been created with the user name "dbadmin" and the password "password".

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

`soldd 'shMem SOLID' dbadmin password`

With this command, all data definitions are listed into one file, `soldd.sql` (the default name). As mentioned earlier, user and role definitions are not listed for security reasons. If the database contains users or roles, they must be appended into this file.

2. All data is extracted with Solid Export. The export results in control files (files with the extension `.ctr`) and data files (files with the extension `.dat`). The default file name is the same as the exported table name. In 16-bit environments, file names longer than eight letters are concatenated. Use the following command line to extract the control and data files for all tables.

`solexp 'shMem SOLID' dbadmin password *`

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

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



Note

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

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

`solcon -eBACKUP 'shMem SOLID' dbadmin password`

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

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

solsql -fSOLDD.SQL 'shMem SOLID' dbadmin password

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

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

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

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

solload 'shMem SOLID' dbadmin password table_name.ctr

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

Batch files that can be used are:

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

Chapter 6. Performance Tuning

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

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

For tips on optimizing SmartFlow data synchronization, see *solidDB SmartFlow Data Replication Guide*.



Tip

The following parameters help you improve database performance or balance performance against safety. These parameters are discussed in more detail in Appendix A, *Server-Side Configuration Parameters*. The *DurabilityLevel* parameter is also discussed in Chapter 6, *Performance Tuning*.

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

6.1 Logging and Transaction Durability

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

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

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

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

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

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

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

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

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



Note

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

The default level of durability is "adaptive".

6.1.2 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. This is acceptable in situations where, for example, you only need to sample data, or where the server can simply operate on the most recent set of data, regardless of the size of that set. As an example, suppose that you have a server that contains statistical data about performance — e.g. which computers experience the heaviest loads at particular times of the day. You might use such information to balance the load on your computers. This information changes over time, and "old" data is less valuable than "new" data. In fact, you might completely discard any data that is more than a week old. If you were to lose the performance and load balancing data, then your system would still function, and within a week you would have acquired a complete set of new data (assuming that you normally discard data older than one week). In this situation, occasional or small data loss is acceptable, and performance may be more important.

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

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

6.1.4 Standards Compliance

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

6.1.5 Limitations on Transaction Durability



Caution

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

If you want to set a maximum delay time before the server writes data, set the *RelaxedMaxDelay* parameter in the `solid.ini` configuration file. For more information about this parameter, see Appendix A in *solidDB Administration Guide*.

6.2 Choosing Transaction Isolation Levels

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

- Read Committed

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

- Repeatable Read

This isolation level is the default isolation level for solidDB databases. It 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. The server's default isolation level is REPEATABLE READ (and therefore the default validation is transaction write set validation).

- **Serializable**

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

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



Note

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

6.2.1 Setting the Isolation Level

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

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

For example:

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

Note that solidDB supports both "transaction-level" and "session-level" isolation level commands. For more details, see the descriptions in solidDB SQL Guide, *Appendix B, Solid SQL Syntax*.

6.3 Understanding Memory Consumption

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



Note

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

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

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

- The accessed tables are also buffered in the main memory.

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

6.3.1 Tuning Your Operating System

Your operating system may store information in:

- real (physical) memory
- virtual memory
- expanded storage
- disk

Your operating system may also move information from one location to another. Depending on your operating system, this movement is called paging or swapping. Many operating systems page and swap to accommodate large amounts of information that do not fit into real memory. However, this takes time. Excessive paging or swapping can reduce the performance of your operating system and indicates that your system's total memory may not be large enough to hold everything for which you have allocated memory. You should either increase the amount of total memory or decrease the amount of database cache memory allocated.

6.3.2 Database Cache

The information managed by solidDB is stored either in memory or on disk. Since memory access is faster than disk access, it is desirable for data requests to be satisfied by access to memory rather than access to disk.

Tuning Cache

Database cache uses available memory to store information that is read from the hard disk. When an application next time requests this information, the data is read from memory instead of from the hard disk. The default value of cache depends on the platform used and can be changed through the *CacheSize* parameter. Increasing the value is recommended when there are several concurrent users.

The following values can be used as a starting point:

- 0.5 MB per each concurrent user of the system

or

- 2-5% of the database size,

whichever is larger



Note

You should increase the value of *Cachesize* carefully. If a value is too large, it leads to poor performance because the server process does not fit completely in memory and therefore swapping of the server code itself occurs. If, on the other hand, the cache size is too small, the cache hit rate remains poor. The symptoms of poor cache performance are database queries that seem to be slower than expected and excessive disk activity during queries. You can verify if the server is retrieving most of the data from disk instead of from RAM by checking the cache hit rate using the command `ADMIN COMMAND 'status'` or by checking the overall cache and file ratio statistics using `ADMIN COMMAND 'perfmon'`. For details on these commands, read Section 3.8.5, “Detailed DBMS Monitoring (Perfmon)” and Section 3.8.1, “Checking Overall Database Status”. Note that the cache hit rate should be better than 95%.

6.3.3 Sorting

By default, solidDB does all sorting in memory. The amount of memory used for sorting is determined by the parameter *SortArraySize* in the `[SQL]` section. If the amount of data to be sorted does not fit into the allocated memory, you may want to increase the value of the parameter *SortArraySize*.

Note that it may seem that the correct setting for the size of the sort array must accommodate the largest expected result set (that cannot be ordered by key values); however, there are some non-intuitive consequences to consider when increasing the sort array size.

If increasing the value of the *SortArraySize* results in slower, rather than faster query times, then it is likely that one of the following behaviors of the Optimizer is involved:

- The *SortArraySize* parameter affects whether indices are used for sorting. If the *SortArraySize* setting is large, the Optimizer is likely to use the sort array for sorting, rather than using the available indices for sorting. If the *SortArraySize* is small, the Optimizer is likely to use the available indices for sorting. In some cases (especially those with small result sets), a small *SortArraySize* setting performs better than a large *SortArraySize* setting.
- The *SortArraySize* parameter affects the way that the Optimizer performs GROUP operations. The Optimizer considers a GROUP operation on non-sorted result sets as an expensive operation. Thus, with smaller settings for the *SortArraySize*, the optimizer causes the result sets to be sorted before performing the GROUP operation. With larger settings for the *SortArraySize*, the GROUP operation tends to proceed without first sorting the result set. In some cases, this can result in slower performance for the larger settings of the *SortArraySize* than for the smaller settings.

Note that for large sorts, or when there is not enough memory to increase the value of *SortArraySize*, you should activate the external sort, which stores intermediate information to disk.

The external disk sort is activated by adding the following section and parameters in the configuration file `solid.ini`:

```
[sorter]
TmpDir_1 = c:\tmp
```

Additional sort directories are added with similar definitions:

```
[sorter]
TmpDir_1 = c:\tmp
TmpDir_2 = d:\tmp
TmpDir_3 = e:\tmp
```

Defining more than one sorter temporary directory on separate physical disks may significantly improve sort performance by balancing the I/O load to multiple disks.

Optimized Sorts

Some queries implicitly require sorting. For example, if the SQL Optimizer chooses a JOIN operation to use the MERGE JOIN algorithm, the result sets to be joined require sorting before the join can occur. You can query the Optimizer's decisions from solidDB using the EXPLAIN PLAN FOR statement. For details, read the description of the EXPLAIN PLAN FOR command in solidDB SQL Guide.

Sorting occurs only if the result set is not returned automatically in the correct order. If the table data is accessed using the primary key or index, then the result set is automatically in the order specified by the index in use. Hence, you can significantly improve server performance by designing primary keys and indices to support the ordering requirements of frequently used, performance-critical queries.

6.3.4 Using In-Memory Database

The solidDB database products use two integrated database engines: one is a traditional disk-based engine and the other is an in-memory database engine allowing to create tables that reside permanently in main memory. Also the indexes created for those tables are stored totally in main memory. When using the in-memory database capability you may choose, for each table, which is the storage for the table: disk or memory. A solidDB server process running in-memory tables is significantly larger than a purely disk-based server process. To evaluate the amount of memory required by the in-memory tables and their indexes, refer to *solidDB In-Memory Database User Guide*.

6.4 Tuning Network Messages

You can improve solidDB performance in reading large result sets by instructing a solidDB server to return several result set rows in one network message. To activate this functionality, you edit one or both of the following parameters in the `[Srv]` section of the solidDB server's `solid.ini` configuration file.

- *RowsPerMessage*. The default value is 10.
- *ExecRowsPerMessage*. The default value is 2.

For more information about these two parameters, see Appendix A, *Server-Side Configuration Parameters*.

6.5 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.

6.5.1 Distributing I/O

Disk contention occurs when multiple processes try to access the same disk simultaneously. To avoid this, move files from heavily accessed disks to less active disks until they all have roughly the same amount of I/O.

Follow these guidelines:

- Use a separate disk for log files.
- Divide your database into several files and place each of these database files on a separate disk. Read Section 4.3.2, “Managing Database Files and Caching (*IndexFile* section)”.
- Consider using a separate disk for the external sorter

It is usually faster to scan a table if the disk file is contiguous on the disk, rather than spread across many non-contiguous disk blocks. To reduce existing fragmentation, you may want to run defragmentation software if one is available on your system. If your database file is growing, you may be able to reduce future file fragmentation by using the configuration parameter *ExtendIncrement*. Increasing the size of this parameter tells the server to allocate larger amounts of disk space when it runs out of space. (Note that this does not guarantee contiguity because the operating system itself may allocate non-contiguous sectors to satisfy even a single request for more space.) As a general rule, larger values of *ExtendIncrement* improve performance slightly, while smaller values keep the database size slightly smaller. See Appendix A, *Server-Side Configuration Parameters*, for more details about *ExtendIncrement*.

6.5.2 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 causes the merge process to start by setting the following parameter in the *General* section of the `solid.ini` file. For example:

```
MergeInterval = 1000
```

Normally the recommended setting is the default value, which is cache size dependent. The default is calculated dynamically from the cache size, so that only part of the cache is used for the Bonsai Tree. If you change the merge interval, be sure that the cache is large enough to accommodate the Bonsai Tree. The longer the merge interval is (i.e. the more data that is stored in memory before being moved to the main storage tree), the larger the cache needs to be.



Note

If the merge interval setting is too big to allow the Bonsai Tree to fit into cache, then it is flushed partially to the disk; this has an adverse affect on performance. Hence, avoid setting merge intervals that are too large. On a diskless system, the Bonsai Tree will fill the available memory and the Diskless server will run out of memory.



Note

Although the server will have higher performance if merge intervals are less frequent (i.e. batch inserts are larger), you may also see less consistent response times. If your highest priority is not overall throughput, but is instead to minimize the longest response time, then you may want to make merge intervals more frequent rather than less frequent. More frequent merges will reduce the worst case delays that interactive users may experience.

For details on detecting and preventing performance problems associated with Bonsai Tree growth, read Section 6.7, “Reducing Bonsai Tree Size by Committing Transactions”.

6.6 Tuning Checkpoints

Checkpoints are used to store a transactionally-consistent state of the database quickly onto the disk.

Checkpoints affect:

- runtime performance
- recovery time performance

Checkpoints cause solidDB to perform data I/O with high priority, which momentarily reduces the run-time performance. This overhead is usually small. As with merge intervals, less frequent checkpoints may mean less frequent, but longer, delays before the system responds to interactive queries. More frequent checkpoints tend to minimize the worst case delays that an interactive user might experience. However, such delays may be more frequent even if they are shorter.

It is possible to control the execution of checkpoints to prevent them from occurring during, for example, periods of high user volume. You may:

- Set configuration parameters in the `solid.ini` file.
 - Set the *CheckpointInterval* parameter in the `solid.ini` configuration file. The default checkpoint interval is every 50000 log writes.
 - Set the *MinCheckpointTime* parameter in `solid.ini`.

For more information about these parameters, see Appendix A, *Server-Side Configuration Parameters*. To learn how to change a parameter value, see Section 4.4, “Managing Server-Side Parameters” in this guide.

- Force a checkpoint by using the **makecp** command. For details on **makecp**, read Section 3.11, “Creating Checkpoints”.

Frequent checkpoints can reduce the recovery time in the event of a system failure. If the checkpoint interval is small, then relatively few changes to the database are made between checkpoints and consequently, few changes need to be made during recovery. To speed up recoveries, create checkpoints frequently; note, however, that the server performance is reduced during the creation of a checkpoint. Furthermore, the speed of checkpoint creation depends on the amount of database cache used; the more database cache is used, the longer the checkpoint creation will take. See Appendix A, *Server-Side Configuration Parameters*, for a description of the use of *CacheSize* parameter. You need to consider these issues when deciding the frequency of checkpoints.

For more details on checkpoints, read Section 3.11, “Creating Checkpoints”. You may also wish to read about transaction logging.

6.7 Reducing Bonsai Tree Size by Committing Transactions

solidDB provides a consistent view of data within one transaction. If a user does not commit a transaction, solidDB keeps an image of the database as it existed at the moment the transaction was started — even if the transaction is a read-only transaction. This is implemented by the multiversioning Solid Bonsai Tree (TM), which stores the newest data in central memory. The new data is merged to the main storage tree as soon as currently active transactions no longer need to see the old versions of the rows.

When other connections perform many write operations, the server must use a large amount of memory to provide a consistent image of the database. If an open transaction remains uncommitted for a long duration of time, solidDB requires more memory; if the amount of memory available is insufficient, then solidDB performs excessive paging or swapping, which slows performance.

To determine whether slow performance is caused by excessive Bonsai Tree growth, you can monitor memory usage and Bonsai Tree size using Operating System-specific and Solid-specific tools.

6.7.1 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, Solid ODBC Driver version 3.50 and Solid JDBC Driver version 2.0 perform an implicit commit after the last open cursor has been closed or dropped. In previous versions, the implicit commit is not available.)

Note that even in autocommit mode, `SELECT` statements are not automatically committed after the data is read. solidDB cannot immediately commit `SELECT`s 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 Solid APIs

Each of these topics is described in the following sections.

Determining Currently Existing Connections

The following Solid commands and files allow you to determine the status of existing connections.

Table 6.1. Determining Command Status

Command/File	Information
ADMIN COMMAND 'ul'	Obtain a list of existing connections.
ADMIN COMMAND 'sta'	Obtain the number of existing connections.
<code>solmsg.out</code>	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 <code>soltrace.out</code> file.
ADMIN COMMAND 'report <i>filename.txt</i> '	Obtain a list of internal variables containing connection and status information.
SolidConsole status screen (from the Menu option, select Administration, Status)	Obtain the number of users and a list of them by clicking the User button.

Determining When Connections Have Committed Transactions

The following Solid commands and files allow you to determine which connections have committed transactions.

Table 6.2. Determining which Connections Have Committed Transactions

Command/File	Information
ADMIN COMMAND 'trace'	Shows if a transaction gets committed at the server
ADMIN COMMAND 'report <i>filename.txt</i> '	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.

Command/File	Information
	To find those statements with active status, look under USER SEARCHES with column 'Act' having a value of 1. If the active status remains at the same Readlevel for a lengthy period of time, this is an indication that the statement has not closed or committed during this interval.

Providing Commit Statements in the Application Code

To make sure every database operation gets committed, be sure to either:

- Execute the statement `COMMIT WORK`
- Call ODBC function `SQLTransact` or `SQLEndTran`.
- Call JDBC method `commit`.

Make sure these operations succeed by checking the return code or by properly catching the possible exception. Be aware how many database connections your application has, when and where they are created, and when the transactions at these connections are committed.

Troubleshooting COMMITs When Using ODBC Driver Manager

When using ODBC Driver Manager and running in autocommit mode, most versions of ODBC Driver Manager regard calls to `SQLTransact` and `SQLEndTran` as redundant and never actually pass them to the driver.

This means that the application program only receives return code 'SUCCESS' from the ODBC Driver Manager, even though no transaction is committed in the database. This situation may go unnoticed. Besides, ODBC Driver Manager, SQL Editor, and SolidConsole, other utilities can also have an open transaction.

Make sure that you are aware of all database connections. Note that each `FETCH` after `COMMIT` (keeping the statement handle alive) also causes a new transaction to start.

6.8 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 6.3. 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.	<ul style="list-style-type: none"> • 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. 	<p>If index definitions are missing, create new indices or modify existing ones to match the indexing requirements of the slow query. For more details, read the section in solidDB SQL Guide titled "Using Indexes to Improve Query Performance".</p> <p>Run the EXPLAIN PLAN FOR statement for the slow query and verify whether the query optimizer is using the indices. For more details, read the description of the EXPLAIN PLAN FOR command in solidDB SQL Guide.</p> <p>If the Optimizer is not choosing the optimal query execution plan, override the Optimizer decision by using optimizer hints. For more details, read "Using Optimizer Hints" in solidDB SQL Guide.</p> <p>Make sure the external sorter is enabled by defining the <i>Sorter.TmpDir</i> configuration parameter. For more details, see the section called "TmpDir_[1...N]".</p>
Slow response time is experienced for all queries. An increase in the number of concurrent users deteriorates the performance more than linearly. When all users are thrown out and then reconnected, performance still does not improve.	Insufficient cache size.	Increase the cache size. Allocate for cache at least 0.5MB per concurrent user or 2-5% of the database size. For more details, read the section called "Tuning Cache".
Slow response time is experienced for all queries and write operations. When all users are thrown out and	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

Symptoms	Diagnosis	Solution
are connected, performance only improves temporarily. The disk is very busy.		read-only transactions) are committed in a timely manner. For more details, read Section 6.7, “Reducing Bonsai Tree Size by Committing Transactions”.
Slow performance during batch write operation as the database size increases. There is an excessive amount of disk I/O.	<ul style="list-style-type: none"> 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. 	<p>Make sure that the autocommit is switched off and the write operations are committed in batches of at least 100 rows per transaction.</p> <p>Modify the primary keys or batch write processes so that write operations occur in the primary key order. For more details, read chapter Optimizing Batch Inserts and Updates in solidDB SQL Guide.</p>
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.

Chapter 7. Managing Network Connections

As a true client/server DBMS, solidDB provides simultaneous support for multiple network protocols and connection types. Both the database server and the client applications can be simultaneously connected to multiple sites using multiple different network protocols.

This chapter describes how to set up network connections for each of the supported platforms.



Note

Some platforms may limit the number of concurrent users to a single solidDB server process even if the Solid license accepts higher limits. Refer to the Release Notes for details that apply to your specific operating system.

7.1 Communication between Client and Server

The database server and client transfer information between each other through the computer network using a communication protocol.

When a database server process is started, it will publish at least one network name that distinguishes it in the network. The server starts to *listen* to the network using the given network name. The network name consists of a communication protocol and a server name.

To establish a connection from a client to a server they both have to be able to use the same communication protocol. The client has to know the network name of the server and often also the location of the server in the network. The client process uses the network name to specify which server it will *connect* to.

This chapter will give you information on how to administer network names.

7.2 Managing Network Names

The network name of a server consists of a *communication protocol* and a *server name*. This combination identifies the server in the network. The network names are defined with the *Listen* parameter in the *[Com]* section of the configuration file. The `solid.ini` file should be located in a solidDB program's working directory or in the directory set by the `SOLIDDIR` environment variable.

A server may use an unlimited number of network names. Note that all components of network names are case insensitive.

Network names are managed in the following ways:

- Using the Protocols page in the solidDB accessed through the Administration window or menu.
- Directly, by editing the server configuration file `solid.ini`.

An example of an entry in `solid.ini` is:

```
[Com]
Listen = tcpip 1313, nmpipe solid
```

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 'solid'. In our example the 'tcpip' and 'nmpipe' are communication protocols, while '1313' and 'solid' are server names. (The conventions for server names depend upon the protocol. A server name may be a name, such as 'solid' or 'chicago_office'. A server name might be a service port number optionally preceded by a node name, such as 'hobbes 1313' or 'localhost 1313'. In some protocols, the server name might simply be a service port number, such as '1313', if the client and server are running on the same computer.)

If the *Listen* parameter is not set in the `solid.ini` file, the environment-dependent defaults are used.



Note

1. When a database server process is started, it publishes the network names that it starts to listen to. This information is also written to a file named `solmsg.out` located in the same directory as the `solid.ini` file.
2. Network names must be unique within one host computer. For example, you cannot have two database servers running, both listening to the same TCP/IP port in one host, but it is possible that the same port number is in use in different hosts. Exceptions to this are the NetBIOS and IPX/SPX protocols, which require that used server names are unique throughout the whole network.

7.2.1 Viewing Supported Protocols for the Server

Because not all protocols are supported in all environments and operating systems, you can view the protocol options available for your server.

To view supported protocols for a server, enter the following command in SolidConsole or Solid SQL Editor (solsql):

ADMIN COMMAND 'protocols'

A list of all available communication protocols is displayed. The command provides the following kind of result set, which contains one row for each supported communication protocol:

```
admin command 'protocols';
  RC TEXT
  -- ----
    0 NetBIOS      nb
    0 NmPipe       np
    0 TCP/IP       tc
3 rows fetched.
```

7.2.2 Viewing Network Names for the Server

Following are ways that you can view network names for the server:

- Select the Status option from the solidDB Administration window or menu and click the Protocols icon to view the network names listed in the Protocols dialog box.
- View the *Listen* parameter in the *[Com]* section in the *solid.ini* file.
- Enter the following command in SolidConsole or Solid SQL Editor (solsql):

ADMIN COMMAND 'parameter com.listen'

A list of all network names for the server is displayed.

7.2.3 Adding and Modifying a Network Name for the Server

Following are ways you can add and edit network names for a server, which consists of a *communication protocol* and a *server name*; for example, *nmpipe solid*.

- Select the Status option from the solidDB Administration window or menu and click the Protocols icon to add or modify the network names in the Protocols dialog box.
- To add network names for the server, enter the following command in SolidConsole or Solid SQL Editor (solsql):

ADMIN COMMAND 'parameter com.listen=*network_name*'

The command returns the new value as the resultset. If the network name entered is invalid, the ADMIN COMMAND statement returns an error. Otherwise the new name is enacted immediately. The changes are written to `solid.ini` at the next checkpoint.

- In `solid.ini`, locate the working directory of your solidDB process and add a new network name or edit an existing one as a part of the *Listen* parameter entry in the *[Com]* section.

Use a comma (,) to separate network names. For example:

```
[Com]
Listen = tcpip 1313, nmpipe solid
```

Be sure to save the changes and to restart the solidDB process to activate the changes.

7.2.4 To Remove a Network Name from the Server

Following are ways you can remove network names for a server, which consists of a *communication protocol* and a *server name*, for example, *nmpipe solid*.

- Select the Status option from the solidDB Administration window or menu and click the Protocols icon to remove the network name in the Protocols dialog box.
- To make the change by updating the `solid.ini` configuration file, locate the working directory of your solidDB process and remove the network name in the *Listen* parameter entry in the *[Com]* section.

Be sure to save the changes and to restart the solidDB process to activate the changes.

When you start the server, if you want to temporarily disable one of the network names listed in the `solid.ini` file, you can disable the network name by using option **-d** after the protocol name in the network name when you start the server. For example:

```
solid tcp -d hobbes 1313
```

This prevents the server from using this network name. This does not change the contents of the `solid.ini` file, so this will have no effect on the server name(s) the next time that the server starts up.

7.2.5 Factory Value for a Network Name

If no network name is specified in the `.ini` file, the server uses a factory preset that is "tcpip 1964". In other words, the server will listen to the TCP/IP port 1964, if no `.ini` file is used.

7.3 Connect Strings for Clients

A network name used by a client is a logical data source name or a data source connect string. A data source connect string consists of a *communication protocol*, a possible set of *special options*, an optional *host computer name* and a *server* name. By this combination, the client specifies the server it will establish a connection to. The communication protocol and the server name must match the ones that the server is using in its network listening name. In addition, most protocols need a specified host computer name if the client and server are running on different machines. All components of the client's network name are case insensitive.

The same format of a connect string for clients applies to both the connect configuration parameters in the `solid.ini` file

and network names used in ODBC and Light Client applications.

The format of a connect string is the following:

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

where options may be any number of:

Table 7.1. Connect String Format

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

Examples:

```
tcp localhost 1315
tcp 1315
```

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

7.3.1 Mapping Logical Data Source Names to Connect Strings

solidDB Clients support Logical Data Source Names. These names can be used for giving a database a descriptive name. This name can be mapped to a data source in three ways:

1. Using the parameter settings in the application's `solid.ini` file.
2. Using the Microsoft Windows operating system's registry settings.
3. Using settings in a `solid.ini` file located in the Windows directory.

This feature is available on all supported platforms. However, on non-Windows platforms, only the first method is available.

A solidDB Client attempts to open the file `solid.ini` first from the directory set by the `SOLIDDIR` environment variable. If the file is not found from the path specified by this variable or if the variable is not set, an attempt is made to open the file from the current working directory.

To define a Logical Data Source Name using the `solid.ini` file, you need to create a `solid.ini` file containing the section `[Data Sources]`. In that section you need to enter the 'logical name' and 'network name' pairs that you want to define. The syntax of the parameters is the following:

```
[Data Sources]
logical_name = connect_string, Description
```

In the description field, you may enter comments on the purpose of this logical name.

For example, assume you want to define a logical name for the application `My_application` and the database that you want to connect is located in a UNIX server using TCP/IP. Then you should include the following lines in the `solid.ini` file, which you need to place in the working directory of your application:

```
[Data Sources]
My_application = tcpip irix 1313, Sample data source
```

When your application now calls the Data Source 'My_application', the solidDB Client maps this to a call to 'tcpip irix 1313'.

On Windows platforms, the registry is typically used to map Data Sources. To setup the registry with a GUI interface, use the Windows Administrative Control Panel "Data Sources (ODBC)".

7.3.2 Default Connect String

When no data source is specified for the connection, the default connect string will be used. The client's default connect string may be defined in the client's configuration file `solid.ini` in the `[Com]` section with the `Connect` parameter. The client's `solid.ini` file should be located in the application program's working directory or in the directory set by the `SOLIDDIR` environment variable. The value of the `Connect` parameter is read by all Solid tool programs and client libraries when no data source is specified for the connection. The client libraries do not need this value if a valid connect string is supplied at run time, or when a standard ODBC driver manager is used.

The following connect line in the `solid.ini` of the application workstation will connect an application (client) using the TCP/IP protocol to a solidDB server running on a host computer named 'spiff' and listening with the name (port number in this case) '1313'.

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

If the `Connect` parameter is not found in the `solid.ini` configuration file, then the client uses the environment-dependent default instead. The defaults for the `Listen` and `Connect` parameters are selected so that the application (client) will always connect to a local solidDB server listening with a default network name. So local communication (inside one machine) does not necessarily need a configuration file for establishing a connection.

7.4 Communication Protocols

A client process and solidDB communicate with each other by using computer networks and network protocols. Supported communication protocols depend on the type of computer and network you are using.

The following paragraphs describe the supported communication protocols and common environments that may be used and also show the required forms of network names for the various protocols.



Note

Depending on your network protocol, there may be relevant communication parameters associated with the protocol. Be sure to use ADMIN COMMAND 'parameter' in the solidDB Query window to find the communication parameters in use. Then you can use ADMIN COMMAND 'describe parameter' to view details on the specific communication parameter. See Chapter 4, *Configuring solidDB* for details on these commands.

7.4.1 Shared Memory

Usually the fastest way two processes can exchange information is to use Shared Memory. This can be used only when solidDB and application processes are both running in the same computer. The Shared Memory protocol uses a shared memory location for moving data from one process to another.

To use the Shared Memory protocol in solidDB, select *ShMem* from the list of protocols in solidDB and enter server name. The server name has to be unique only in this computer.

Table 7.2. Format Used in the `solid.ini` File

Server	<code>Listen = shmем servername</code>
Client	<code>Connect = shmем servername</code>



Note

Server names must be character strings less than 128 characters long.

7.4.2 TCP/IP

When starting a server using the TCP/IP protocol, you must reserve a port number for it. You will find reserved port numbers in the `/etc/services` file of your system. Select a free number greater than 1024 since smaller numbers are usually reserved for the operating system.

To use the TCP/IP protocol, select TCP/IP in the list of protocols in solidDB and enter a non-reserved port number.

Table 7.3. Format Used in the `solid.ini` File

Server	<code>Listen = tcpip server_port_number</code>
--------	------------------------------------------------

Client	<code>Connect = tcpip [host_computer_name] server_port_number</code>
--------	----------------------------------------------------------------------

For example

```
Listen = tcp 1315
Connect = tcpip accounting_dept_server 1315
```



Note

1. If the server is running in the same computer with the client program, the host computer name need not be specified. The client computer must have the used host name listed in its `/etc/hosts` file or it must be recognized by the DNS (Domain Name Server). You can also give the host computer's TCP/IP address in dotted decimal format (for example, 194.53.94.97) instead of its host name.
2. On Windows and UNIX, the TCP/IP protocol is usually included in the operating system. On other environments (like VAX/VMS) the TCP/IP software needs to be installed on the system. For a list of supported TCP/IP software, contact Solid Information Technology Ltd at: <http://www.solidDB.com>
3. The local loopback interface address, 127.0.0.1, is the default address when a client attempts to open a TCP/IP connection without specifying a hostname.
4. Using option **-iip_address** or **-ihost_name**, the solidDB listens only to the specified IP address or host name. This is useful in multi-homed systems that support many TCP/IP interfaces (or have multiple ip-addresses). For example, a server with the following setting in `solid.ini` accepts connection requests only from inside the same machine, either referred by IP address 127.0.0.1 or with the name 'localhost', if the DNS is correctly configured:

```
[com]
Listen = tcp -i127.0.0.1 1313
```

Note that DNS entries can be used instead of IP addresses, for example:

```
[com]
```

```
Listen = tcp -ilocalhost 1313
```

5. Using option `-i127.0.0.1`, which starts the server to listen only to a local loopback connection, allows TCP/IP listening with a desktop license. To enable TCP/IP usage with desktop licenses, all entries in `solid.ini` have to be edited to include `-i`. Note that default listening of port 1313 (without `solid.ini`) works automatically.

7.4.3 UNIX Pipes

The UNIX domain sockets (UNIX Pipes) are typically used when communicating between two processes running in the same UNIX machine. UNIX Pipes usually have a very good throughput. They are also more secure than TCP/IP, since Pipes can only be accessed from applications that run on the computer where the server executes.

When starting a server using UNIX Pipes, you must reserve a unique listening name (inside that machine) for the server, for instance, 'solid'. Because UNIX Pipes handle the UNIX domain sockets as standard file system entries, there is always a corresponding file created for every listened pipe. In solidDB's case, the entries are created under the path `/tmp`. Our example listening name 'solid' creates the directory `/tmp/solunp_SOLID` and shared files in that directory. The `/tmp/solunp_` is a constant prefix for all created objects while the latter part ('SOLID' in this case) is the listening name in upper case format.

Table 7.4. Format Used in the `solid.ini` File

Server	<code>Listen = upipe server_name</code>
Client	<code>Connect = upipe server_name</code>



Note

1. Server and client processes must run in the same machine in order to use UNIX Pipes for communication.
2. The server process must have "write" permission to the directory `/tmp`.

3. The client that is accessing UNIX Pipes must have "execute" permission on the directory `/tmp`.
4. The directory `/tmp` must exist.
5. UNIX Pipes cannot be used in Caldera/SCO UNIX.

7.4.4 Named Pipes

Named Pipes is a protocol commonly used in the Microsoft Windows operating systems.

Table 7.5. Format Used in the `solid.ini` File

Server	<code>Listen = nmpipe server_name</code>
Client	<code>Connect = nmpipe [host_computer_name] server_name</code>



Note

1. The server names must be character strings at most 50 characters long.
2. If the server is running in the same computer with the application program, the host computer name should not be specified.
3. In order to connect to the solidDB for Windows through Named Pipes, the user must have at least the same rights as the user who started the server. For example if an administrator starts the server, then only users with administrator's rights are able to connect to the server through Named Pipes. Similarly, if a user with normal user's rights starts the server, then all users with equal or greater rights are able to connect the server through Named Pipes. If a user doesn't have proper rights, Solid Communication Error 21306 message will be given.
4. It is not recommended to use the Named Pipes communication from Solid Remote Control. The asynchronous nature of Solid Remote Control communication may cause problems with Named Pipes.

Note that you may use either "nmpipe" or "nmp" to specify the named pipes protocol.

7.4.5 NetBIOS

The NetBIOS protocol is commonly used in the Microsoft Windows operating systems.

To use NetBIOS protocol, select NetBIOS in the list of available protocols in solidDB and enter a non-reserved server name.

Table 7.6. Format Used in the `solid.ini` File

Server	<code>Listen = netbios [aLANA_NUMBER] server_name</code>
Client	<code>Connect = netbios [aLANA_NUMBER] server_name</code>



Note

1. The server name must be a character string at most 16 characters long. It may not begin with an asterisk (*).
2. In the above format, the optional `-aLANA_NUMBER` parameter is used to override the default value of the LANA number.
3. In Windows, the available LANA numbers can be checked using the Network Setup found in the Control Panel. The default value 0 may not be generally very good. You should choose the one(s) where the protocol stack matches the other computers you are using. The LANA number (Network Route: Nbf → Elnk3 → Elnk31) that uses NetBEUI as a transport usually functions quite smoothly when used for solidDB communication.
4. The server names have to be unique in the whole network. Establishing a connection or starting the listener using the NetBIOS protocol may be somewhat slow because of the checks needed for uniqueness.
5. Solid products use all available LANA numbers by default. This makes it unnecessary to specify explicitly which LANA number the application or solidDB should use. For backward compatibility, the `-aLANA_NUMBER` parameter remains available.

7.4.6 DECnet

The DECnet protocol is used to connect to solidDB running on an OpenVMS system. To use this protocol in Windows you need to have PATHWORKS 32 installed on your client computer.

To use the DECnet protocol, select DECnet in the list of protocols in SolidConsole and enter a non-reserved server name.

Table 7.7. Format Used in the `solid.ini` File

Server	<code>Listen = decnet <i>server_name</i></code>
Client	<code>Connect = decnet <i>node_name server_name</i></code>



Note

To establish a connection, the DECnet node name of the server machine is configured to your node database. The node name can be given either as a node number such as '1.1' or as a node name such as 'VAX1'.

7.4.7 A Summary of Protocols

The following tables summarize the possible operating systems and required forms for network names for the various communication protocols.



Note

The following tables contain the protocols and operating systems that were available when this guide was printed. For an updated list, refer to the Solid Website at: <http://www.solidDB.com>.

Table 7.8. solidDB Protocols and Network Names

Protocol	Server OS	Network name in <code>solid.ini</code> file
Shared Memory	Windows	<code>Listen = shm <i>server</i></code>
NetBIOS	Windows	<code>Listen = netbios <i>server</i></code>

Protocol	Server OS	Network name in <code>solid.ini</code> file
Named Pipes	Windows	Listen = <code>nmpipe server</code>
TCP/IP	Windows, UNIX, VxWorks	Listen = <code>tcpip port</code>
UNIX Pipes	UNIX	Listen = <code>upipe server</code>

Table 7.9. Application Protocols and Network Names

Protocol	Server OS	Network name in <code>solid.ini</code> file
Shared Memory	Windows	Connect = <code>shmem server</code>
NetBIOS	Window	Connect = <code>netbios server</code>
Named Pipes	Windows	Connect = <code>nmpipe [host] server</code>
TCP/IP	Windows, UNIX, VxWorks	Connect = <code>tcpip [host] port</code>
UNIX Pipes	UNIX	Connect = <code>upipe server</code>
DECnet	Windows ¹	Connect = <code>decnet host server</code>

1) Requires Digital PATHWORKS 32 for Microsoft Windows.

7.5 Logical Data Source Names

solidDB Clients support Logical Data Source Names. These names can be used for giving a database a descriptive name. This name can be mapped to a network name in three ways:

1. Using the parameter settings in the application's `solid.ini` file.
2. Using the Microsoft Windows operating system's registry settings.
3. Using settings in a `solid.ini` file located in the Windows directory.

This feature is available on all supported platforms. However, on non-Windows platforms, only the first method is available.

A solidDB Client attempts to open the file `solid.ini` first from the directory set by the `SOLIDDIR` environment variable. If the file is not found from the path specified by this variable or if the variable is not set, an attempt is made to open the file from the current working directory.

To define a Logical Data Source Name using the `solid.ini` file, you need to create a `solid.ini` file containing the section `[Data Sources]`. In that section you need to enter the *logical name* and *network name* pairs that you want to define. The syntax of the parameters is the following:

```
[Data Sources]
logical_name = network_name, Description
```

In the description field, you may enter comments on the purpose of this logical name.

For example, assume you want to define a logical name for the application *My_application* and the database that you want to connect is located in a UNIX server using TCP/IP. Then you should include the following lines in the *solid.ini* file, which you need to place in the working directory of your application:

```
[Data Sources]
My_application = tcpip irix 1313, Sample data source
```

When your application now calls the Data Source *My_application*, the solidDB Client maps this to a call to 'tcpip irix 1313'.

On Windows platforms, the registry can be used to map Data Sources. These follow the standards of mapping ODBC Data Sources on a system.

In Windows, a Data Source may be defined in the Windows Registry. The entry is searched from the path *software\odbc\odbc.ini*

1. first under the root *HKEY_CURRENT_USER* and if not found,
2. under the root *HKEY_LOCAL_MACHINE*.

The order of resolving a Data Source name in Microsoft Windows systems is the following:

1. Look for the Data Source Name from the *solid.ini* file in the current working directory, under the section *[Data Source]*
2. Look for the Data Source Name from the following registry path *HKEY_CURRENT_USER\software\odbc\odbc.ini\DSN*
3. Look for the Data Source Name from the following registry path *HKEY_LOCAL_MACHINE\software\odbc\odbc.ini\DSN*

If an application uses normal ODBC Data Sources, the network name is mapped normally using the methods that are provided in the ODBC Driver Manager.

Chapter 8. Diagnostics and Troubleshooting

This chapter provides information on the following solidDB diagnostic tools:

- Network trace facility used to trace the server communication
- Ping facility used to trace client communication

You can use these facilities to observe performance, troubleshoot problems, and produce high quality problem reports. These reports let you pinpoint the source of your problems by isolating them under product categories (such as Solid ODBC API, Solid ODBC Driver, Solid JDBC Driver, etc.).

You may also want to read Section 3.8.5, “Detailed DBMS Monitoring (Perfmon)”, which discusses various monitoring techniques including the perfmon command.

8.1 Tracing Communication between Client and Server

solidDB provides the following tools for observing the communication between an application or an external application (if using AcceleratorLib) and a database server:

- the Network Trace facility
- the Ping facility

You can use these tools to analyze the functionality of the networking between an application and solidDB. The network trace facility should be used when you want to know why a connection is not established to solidDB. The ping facility is used to determine how fast packets are transferred between an application and a database server.

8.1.1 The Network Trace Facility

Network tracing can be done on the solidDB computer, on the application computer or on both computers concurrently. The trace information is written to the default trace file or file specified in the *TraceFile* parameter.

The default name of the output file is `soltrace.out`. This file will be written to the current working directory of the server or client depending on which end the tracing is started.

The file contains information about:

- loaded DLLs
- network addresses
- possible errors

The Network Trace facility is turned on by editing the configuration file:

```
[Com]
Trace = {Yes|No}
; default No
TraceFile = file_name
; default soltrace.out
```

or by using the environment variables *SOLTRACE* and *SOLTRACEFILE* to override the definitions in the configuration file. Setting of *SOLTRACE* and *SOLTRACEFILE* environment variables have the same effect as the parameters *Trace* and *TraceFile* in the configuration file.



Note

Defining the *TraceFile* configuration parameter or the *SOLTRACEFILE* environment variable automatically turns on the Network trace facility.

A third way to turn on the Network trace facility is to use the option *-t* and/or *-ofilename* as a part of the network name. The option *-t* turns on the Network trace facility. The option *-o* turns on the facility and defines the name of the trace output file.

Example 8.1. Defining Parameter Trace in the Client-Side Configuration File

```
[Com]
Connect = nmp SOLID
Listen = nmp SOLID
Trace = Yes
```

Example 8.2. Defining Environment Variables

```
set SOLTRACE = Yes
```

or

```
set SOLTRACEFILE = trace.out
```

Example 8.3. Using Network Name Options

```
[Com]
Connect = nmp -t solid
Listen = nmp -t solid
```

or

```
[Com]
Connect = nmp -oclient.out solid
Listen = nmp -oserver.out solid
```

Example 8.4. 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'.
```

8.1.2 The Ping Facility

The Ping facility can be used to test the performance and functionality of the networking. The Ping facility is built into all solidDB client applications and is turned on with the network name option **-plevel**.

The output file will be written to the current working directory of the computer where the parameter is given. The default name of the output file is `soltrace.out`.

Clients can always use the Ping facility at level 1. Levels 2, 3, 4 or 5 may only be used if the server is set to use the Ping facility at least at the same level.

The Ping facility levels are:

Table 8.1. Ping Facility Levels

Setting	Function	Description
0	no operation	do nothing, default
1	check that server is alive	exchange one 100 byte message
2	basic functional test	exchange messages of sizes 0.1K, 1K, 2K..30K, increment 1K
3	basic speed test	exchange 100 messages of sizes 0.1K, 1K, 8K and display each sub-result and total time
4	heavy speed test	exchange 100 messages of sizes 0.1K, 1K, 2K, 4K, 8K, 16K and display each sub-result and total time
5	heavy functional test	exchange messages of sizes 1..30K, increment 1 byte

**Note**

If a solidDB client does not have an existing server connection, you can use the `SQLConnect ()` function with the connect string `-p1` option (ping test, level 1) to check if solidDB is listening in a certain address. Without logging into solidDB, `SQLConnect ()` can then check the network layer and ensure solidDB is listening. When used in this manner, `SQLConnect ()` generates error code 21507, which means the server is alive.

Example 8.5. Running Ping Facility at Level 1

The client turns on the Ping facility by using the following network name:

```
nmp -p1 -oping.out SOLID
```

This runs the Ping facility at the level 1 into a file named `soltrace.out`. This test checks if the server is alive and exchanges one 100 byte message to the server.

After the Ping facility has been run, the client exits with the following message:

```
SOLID Communication return code xxx: Ping test successful/failed,  
results are in file FFF.XX
```

Example 8.6. How the Listen Parameter Restricts the Use of Ping Facility

If the server is using the following listen parameter, applications can run the Ping facility at levels 1, 2, and 3, but not 4 and 5.

```
[Com]  
Listen = nmp -p3 SOLID
```



Note

Ping clients running at level greater than 3 may cause heavy network traffic and may slow down any application that is using the network, including any ordinary SQL clients connected to the same solidDB.

8.2 Problem Reporting

solidDB offers sophisticated diagnostic tools and methods for producing high quality problem reports with very limited effort. Use the diagnostic tools to capture all the relevant information about the problem.

All problem reports should contain the following files and information:

- `solid.ini`
- license number
- `solmsg.out`
- `solerror.out`
- `soltrace.out`
- `ssdebug.out`
- problem description
- steps to reproduce the problem
- all error messages and codes
- contact information, preferably email address of the contact person

8.3 Problem Categories

Most problems can be divided into the following categories:

- Solid ODBC API

- Solid ODBC or JDBC Driver
- UNIFACE driver for solidDB
- Communication problems between the application or an external application (if using AcceleratorLib) and solidDB.
- Problems with disk block integrity

The following pages include detailed instructions to produce a proper problem report for each problem type. Please follow the guidelines carefully.

8.3.1 Solid ODBC API Problems

If the problem concerns the performance of a specific Solid ODBC API or SQL statement, you should run SQL info facility at level 4 and include the generated `soltrace.out` file into your problem report. This file contains the following information:

- create table statements
- create view statements
- create index statements
- SQL statement(s)

8.3.2 Solid ODBC Driver Problems

If the problem concerns the performance of Solid ODBC Driver, please include the following information:

- Solid ODBC Driver name, version, and size
- ODBC Driver Manager version and size

If the problem concerns the cooperation of solidDB and any third party standard software package, please include the following information:

- Full name of the software
- Version and language
- Manufacturer
- Error messages from the third party software package

Use ODBC trace option to get a log of the ODBC statements and include it in your problem report.

8.3.3 Solid JDBC Driver Problems

If the problem is related to the Solid JDBC Driver, please include the following information in your problem report:

- Exact version of JDK or JRK used
- Name, size, and date of the SOLIDDriver class package
- Contents of `DriverManager.setLogStream(someOutputStream)` output, if available
- Call stack (that is, `Exception.printStackTrace()` output) of the application, if an exception has occurred in the application

8.3.4 UNIFACE Driver for solidDB Problems

If the problem concerns the performance of Solid UNIFACE Driver, please include the following information:

- Solid UNIFACE Driver version and size
- UNIFACE version and platform
- Contents of the UNIFACE message frame
- Error codes from the driver, \$STATUS, \$ERROR
- All necessary files to reproduce the problem (TRXs, SQL scripts, USYS.ASN etc.)

8.3.5 Communication between a Client and Server

If the problem concerns the performance of the communication between a client and server use the Network trace facility and include the generated trace files into your problem report. Please include the following information:

- Solid communication DLLs used: version and size
- Other communication DLLs used: version and size
- Description of the network configuration

8.3.6 Database Disk Block Integrity

If the problem concerns the database disk block integrity, check the integrity by starting solidDB database with the **-x testblocks** parameter. This option will check the disk block integrity and produce a report in the `ssdebug.out` file.

Appendix A. Server-Side Configuration Parameters

By managing the configuration parameters of your solidDB, you can modify the environment, performance, and operation of the server. The configuration parameters are stored in the `solid.ini` configuration file and are read when the server starts.

Generally, the factory value settings offer the best performance and operability, but in some special cases modifying a parameter will improve performance. You can change the parameters in the following ways:

- Manually editing the configuration file `solid.ini`. Since the file is only read when the server is started, changes to a parameter value in the `solid.ini` file do not take effect until the next time that the server is started.
- Entering the command

```
ADMIN COMMAND 'parameter name=value'
```

The first part of this appendix focuses on the `solid.ini` file, and describes the proper format for parameter values in that file.

The second part of this appendix describes how to use an ADMIN COMMAND to change the value of a parameter dynamically.

The remainder of this appendix describes the parameters themselves, including the valid range of values and the factory values.



Note

Parameters for some options, such as the CarrierGrade option, may be described in the manual for that option rather than in this administrator guide.

A.1 Setting Parameters through the `solid.ini` Configuration File

When the solidDB is started, it attempts to open the configuration file `solid.ini`. If the file does not exist, solidDB will use the factory values for the parameters. If the file exists, but a value for a particular parameter

is not set in the `solid.ini` file, solidDB will use a factory value for that parameter. The factory values depend on the operating system you are using.

By default, the server looks for the `solid.ini` file in the current working directory, which is normally the directory from which you started the server. If you would like to specify a different directory to be used as the current working directory, then use the `"-c"` command line option. (For more details about command line options, see Appendix C, *solidDB Command Line Options*.) If you want to specify a different directory for the `solid.ini` file, you can set the `SOLIDDIR` environment variable to specify the location of the `solid.ini` file. When searching for the file, the solidDB uses the following precedence (from high to low):

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

A.1.1 Rules for Formatting the `solid.ini` File

The configuration file `solid.ini` is an ASCII file with line breaks.

The `solid.ini` configuration file is divided into sections. Each section contains a group of one or more loosely-related parameters. Each section has a name, and that name is delimited with square brackets, e.g.

[SQL]

Within each section are the parameters. Parameters are specified in the following format:

```
param_name=param_value
```

for example:

```
Listen=tcp 127.123.45.156 1313
DurabilityLevel=2
```

Blank spaces around the equals sign are allowed but not required. The following are equivalent:

```
DurabilityLevel=2
DurabilityLevel = 2
```

If you omit the parameter value, then the server will use the factory value. For example:

```
; Use the factory value
DurabilityLevel=
```

If you omit the parameter value and the equals sign, you get an error message.

Every parameter must be under a section header. If you put a parameter before any section header, you get an error message indicating that there is an unrecognized entry in the section named "<no section>".

Section names can be repeated. For example:

```
[Index] BlockSize=2048
[Com]
...
[Index]
CacheSize=8m
```

However, repeating sections names makes it more difficult for users to keep the file up-to-date and consistent, so we do not recommend doing this.

Parameter names can also be repeated (you won't get a warning message), but this is very strongly discouraged. The last occurrence of the parameter in the file takes the precedence.

The `solid.ini` file can contain comments, which must begin with a semicolon.

```
; This is a valid comment.
```

You can also put a comment on the same line as a parameter.

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

Below is a simple example of part of a `solid.ini` file that contains a section heading, a parameter, and a comment:

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

[Com]

...

There are a few cases where two or more sections have parameters with the same name. Therefore, you must be careful to place each parameter in the correct section.

Most sections and parameters are optional. You do not need to specify a value for every parameter in every section, and in fact you can omit entire sections. If you omit a parameter(s), the server will use the factory value. Later in this appendix, we list each section, each parameter name, the factory value for that parameter, and a description of the purpose and valid range of values for that parameter.

The server checks each entry in the `solid.ini` file. If the entry is not a comment, the server checks that the combination of section name and parameter name is valid. If you have invalid entries in the file, the server will display an error message in the `solmsg.out` file; if the server is running as a foreground process, the message will also be displayed on the console. The message will be similar to one of the following:

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

You will see this message if you have entries that fit the proper form, but which do not have the pre-defined section names and parameter names. For example, you would get this message if you had a `solid.ini` file like the following:

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

```
This has an invalid section name.  
[NoSuchSectionName]
```

The message for the first of these errors would be similar to:

```
Warning: Unrecognized entry 'Logging.NoSuchParam' in inifile.
```

2. Warning: Illegal entry in inifile: <whole illegal line>

The server will display this message if a line could not be recognized as a section header, parameter name, comment, or blank line. You may see this message if you have entries that are not in the proper form. For example, you will see this message if your `solid.ini` file contains something like the following:

```
; This text was intended to be a comment  
but we forgot to precede part of it with a semicolon.
```

3. Warning: 1 unrecognized or illegal entry in '<inifilename>'

or

Warning: <number> unrecognized or illegal entries in '<inifilename>'.

After the server has finished processing the `solid.ini` file, it will list the total number of errors detected.

4. Warning: Unregistered parameter <section>.<parameter> is used.

If this error occurs, it is a sign of a possible problem inside the server itself. If you see this error, please report it to Solid.

Note that the server does not necessarily display an error message if you use an invalid value for a parameter. The server may simply use the factory value without issuing an error message.

The `solid.ini` parameter file is checked only when the server starts. If you edit it after the server starts, the server will not see the changes until the next time that the server starts.



Caution

If you make changes to the `solid.ini` file AND you make changes to parameters in the server by using an ADMIN COMMAND, the behavior is unpredictable. While the server is running, you can safely change the `solid.ini` file OR make changes to server values using the ADMIN COMMAND, but you should not do both during the same "run" of the server.

A summary of the rules is below:

- Section name is in the format
`[section-name]`
- The same section name may be used several times (however, this is not recommended).
- Each parameter is set in a separate line.
- Entries in the files may be preceded with blanks.

- If the first non-blank character is the comment character, then the whole line is ignored (i.e. it is treated as a comment line).
- The comment character is the semicolon (;).
- Comments may follow other entries that are in the same line.
- Lines that have no characters, or that have only blank characters, are ignored.

A.1.1.1 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 INI file or an ADMIN COMMAND:

- The section and parameter names are not case-sensitive.
- The string values are not case-sensitive.
- In most cases, units are not case-sensitive. For example, to specify that the units are in megabytes, you may use any of the following: m, M, MB, mb, Mb, or mB. Some units (e.g. time units 's' (seconds) and 'ms' (milliseconds)) are case sensitive and such cases are documented.
- The syntax for general parameter value setting is:

```
param_name [space characters] = [space characters] value_literal
```

The syntax for the value is

```
value_literal [space characters] unit_of_measure
```

where

param_name is the parameter name. When this is used in an ADMIN COMMAND, the name should be the full parameter name, including the section name, for example, *Logging.DurabilityLevel*. When this is used in the *solid.ini* file, it should NOT include the section name, since the parameter should already be listed under the appropriate section header.

value_literal is the value to be assigned to the parameter. This is usually a literal, such as the number 12, or the string "tcp MyServer2 1315". If you give no value, the parameter will be set to its startup value. If you assign a parameter value with an asterisk (*), the parameter will be set to its factory value. Note that string literals should normally be in double quotes if they are used in an ADMIN COMMAND.

unit_of_measure is the unit of measure, for example *MB* for megabytes or *ms* for milliseconds.

[space characters] represents places where spaces are allowed but not required. Spaces around the equals sign are optional. Spaces between the value and the unit of measure are optional.

For example, allowed forms include:

```
CacheSize=32M
cachesize=32m
CacheSize = 32 m
etc.
```

A.2 Changing Parameters through an ADMIN COMMAND

Most parameters can be changed with an ADMIN COMMAND:

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

The *param_name* and *value* generally follow the rules specified in Section A.1.1.1, “Format of Configuration Parameter Names and Values”.



Note

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

Note that the *param_name* in an ADMIN COMMAND (unlike in the *solid.ini* file) must include the section name and the parameter name, separated by a period character. For example, to set the value of the *DurabilityLevel* parameter, which is part of the *[Logging]* section, issue a command like:

```
ADMIN COMMAND 'parameter Logging.DurabilityLevel=1';
```

When the value of a parameter is changed with an ADMIN command, the change may or may not apply immediately, and may or may not apply the next time that the server is started. If a parameter value is written to the *solid.ini* file, then it will take effect the next time that the server starts. If the temporary option is used, then the value will affect the server's current behavior, but will not affect the server when it restarts. In

some cases, changing a parameter may take effect immediately AND be written to the `solid.ini` file so that it also applies the next time that the server starts. See the explanations of Access Mode below.

Access Mode

The tables later in this appendix list the "Access Mode" for each parameter. The Access Mode indicates whether the parameter can be changed dynamically (via an ADMIN COMMAND), and when the change takes effect. The possible Access Modes are:

- RO (read-only): the value cannot be changed; the current value is always identical to the startup value.
- RW: can be changed via an ADMIN COMMAND, and the change takes effect immediately.
- RW/Startup: can be changed via an ADMIN COMMAND, and the change takes effect the next time that the server starts.
- RW/Create: can be changed via an ADMIN COMMAND, and the change applies when a new database is created.

Saving Parameter Changes

Unless the option temporary is used, all the changes made to the parameters will be saved in the `solid.ini` file at the next checkpoint. The saving may be also expedited with the command:

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

By default, the command rewrites the default `solid.ini` file. By using the `file_name` option, the output can be directed to a different location.

A.3 Descriptions of Configuration Parameters

There is one table below for each section of the `solid.ini` file. The sections (and tables) are:

- Accelerator
- Cluster
- Com
- General
- HotStandby (discussed in *solidDB High Availability User Guide*)

- IndexFile
- Logging
- MME
- Sorter
- SQL
- Srv
- Synchronizer

Most parameters in most sections apply to all solidDB products (EmbeddedEngine, FlowEngine, and BoostEngine). The sections that do not apply to all products are listed below:

- The *MME* section applies only to BoostEngine.
- The *Synchronizer* section applies only to solidDB SmartFlow capability, which is available in FlowEngine and BoostEngine.
- The *HotStandby* section only applies to the CarrierGrade (HotStandby) option.

The descriptions of a few individual parameters specify that those parameters (or some specific settings of those parameters) apply only to a particular option or product. Each of these exceptions is documented in the description of the parameter itself.

A.4 Accelerator Section

Table A.1. Accelerator Parameters

[Accelerator]	Description	Factory Value
<i>ImplicitStart</i>	If set to yes, this parameter starts solidDB automatically as soon as the ODBC API function <code>SQLConnect</code> is called in a user application. If set to no, solidDB must be explicitly started with a call to the Control API function <code>SSC-StartServer</code> .	yes

A.5 Cluster Section

Table A.2. Cluster Parameters

<i>[Cluster]</i>	Description	Factory Value	Access Mode
<i>ReadMostlyLoadPercentAtPrimary</i>	Percentage of read load directed to the Primary	50	RW/Startup

A.6 Communication Section

Table A.3. Communication Parameters

<i>[Com]</i>	Description	Factory Value	Access Mode
<i>Listen</i>	Defines the network name for a server. When a solidDB database server process is started, it will publish at least one network name that distinguishes it in the network. The server can then start to listen to the network using the given network name. The network name consists of a communication protocol and a server name. For more details, read Chapter 7, <i>Managing Network Connections</i> .	tcp 1964	RW
<i>MaxPhysMsgLen</i>	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
<i>RConnectLifetime</i>	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	60 Unit: 1 second	RW/Startup
<i>RConnectPoolSize</i>	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	10	RW/Startup

[Com]	Description	Factory Value	Access Mode
	until there is room in the pool. Valid values range from 1-1000		
<i>RConnectRPCTimeout</i>	RPC timeout for remote connections. Default is 0 (no timeout).	0. Unit 1 milli-second	RW/Startup
<i>ReadBufSize</i>	Sets the buffer size in bytes for the data read from the network	OS dependent	RW/Startup
<i>SocketLinger</i>	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 <code>close()</code> is issued. The parameter affects all server side connections, including Flow and HSB.	Yes	RW/Startup
<i>SocketLingerTime</i>	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
<i>TcpKeepAlive</i>	<p>This parameter can only be used for Linux, HP-UX, Solaris and QNX platforms. On other platforms, the parameter has no effect.</p> <p>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.</p> <p>See also parameters <i>TcpKeepAliveIdleTime</i>, <i>TcpKeepAliveProbeCount</i> and <i>TcpKeepAliveProbeInterval</i>.</p>	No	RW/Startup
<i>TcpKeepAliveIdleTime</i>	<p>This parameter can only be used for Linux, HP-UX, Solaris and QNX platforms. On other platforms, the parameter has no effect.</p> <p>This parameter controls the TCP_KEEPIPLE socket option. If the SO_KEEPALIVE option is enabled with</p>	7200	RW/Startup

[Com]	Description	Factory Value	Access Mode
	<p>the <i>TcpKeepAlive</i> 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_KEEPIPLE specifies the number of seconds before TCP will send the initial keepalive probe.</p> <p>See also parameters <i>TcpKeepAlive</i>, <i>TcpKeepAliveProbeCount</i> and <i>TcpKeepAliveProbeInterval</i>.</p>		
<i>TcpKeepAliveProbeCount</i>	<p>This parameter can only be used for Linux, HP-UX, Solaris and QNX platforms. On other platforms, the parameter has no effect.</p> <p>This parameter controls the TCP_KEEPCNT socket option. If the SO_KEEPALIVE option is enabled with the <i>TcpKeepAlive</i> 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.</p> <p>See also parameters <i>TcpKeepAlive</i>, <i>TcpKeepAliveIdleTime</i> and <i>TcpKeepAliveProbeInterval</i>.</p>	9	RW/Startup
<i>TcpKeepAliveProbeInterval</i>	<p>This parameter can only be used for Linux, HP-UX, Solaris and QNX platforms. On other platforms, the parameter has no effect.</p> <p>This parameter controls the TCP_KEEPIPLE socket option. If the SO_KEEPALIVE option is enabled with the <i>TcpKeepAlive</i> 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</p>	75	RW/Startup

[Com]	Description	Factory Value	Access Mode
	<p>keepalive probe for a certain number of times before a connection is considered to be broken. The <code>TCP_KEEPINTVL</code> option specifies the number of seconds to wait before retransmitting a keepalive probe.</p> <p>See also parameters <i>TcpKeepAlive</i>, <i>TcpKeepAliveIdleTime</i> and <i>TcpKeepAliveProbeCount</i>.</p>		
<i>Trace</i>	If this parameter is set to yes, trace information on network messages for the established network connection is written to a file specified with the <i>TraceFile</i> parameter. The factory value for the <i>TraceFile</i> parameter is <code>soltrace.out</code> .	no	RW/Startup
<i>TraceFile</i>	If the <i>Trace</i> parameter is set to yes, trace information on network messages is written to a file specified with this <i>TraceFile</i> parameter.	<code>sol-trace.out</code> (written to the current working directory of the server or client depending on which end the tracing is started)	RW/Startup
<i>WriteBufSize</i>	Sets the buffer size in bytes for the data written into the network	OS dependent	RW/Startup

A.7 General Section

Table A.4. General Parameters

[General]	Description	Factory Value	Access Mode
<i>BackupBlockSize</i>	Block size for backup file writing	64 KB Unit: 1 byte k=KB	RW/Startup

[General]	Description	Factory Value	Access Mode
<i>BackupCopyIniFile</i>	If set to yes, <code>solid.ini</code> file will be copied to the backup directory	yes	RW/Startup
<i>BackupCopyLog</i>	If set to yes, backup operation will copy log files to the backup directory	yes	RW/Startup
<i>BackupCopySolmsgOut</i>	If set to yes, <code>solmsg.out</code> file is copied to the backup directory	yes	RW/Startup
<i>BackupDeleteLog</i>	If set to yes, old log files will be deleted after backup operation	yes	RW/Startup
<i>BackupDirectory</i>	<p>Makes a backup of the database, log files, and the configuration file <code>solid.ini</code>, using the factory value 'backup' or a given name. For example, <code>BackupDirectory=abc</code>, creates a backup on directory 'abc'.</p> <p>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.</p> <p>All directory definitions are relative to the solidDB working directory unless the full path is provided.</p> <p>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.</p>	'backup' directory	RW/Startup
<i>BackupStepsToSkip</i>	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
<i>CheckpointDeleteLog</i>	If this parameter is set to yes, then the server deletes the transaction log file(s) after each successful checkpoint. This saves disk space, but makes it impossible to recover data by rolling forward the logs.	no	RW/Startup

[General]	Description	Factory Value	Access Mode
	<p>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.</p> <p>You should only set <i>CheckpointDeleteLog</i> to yes if your database has data that you are willing to risk losing (e.g. test data created during development). See also the <i>BackupDeleteLog</i> parameter.</p> <p>NOTE: If you are using the CarrierGrade (HotStandby) option, and if you set <i>CheckpointDeleteLog=Yes</i> on the Primary server, then the server deletes only the logs that are already acknowledged by Secondary. For example, if the Secondary is down and the Primary is in PRIMARY ALONE state, then the Primary will keep the logs even after the data has been checkpointed on the Primary.</p>		
<i>CheckpointInterval</i>	<p>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: <i>MinCheckpointTime</i>. (Note that <i>CheckpointInterval</i> and <i>MinCheckpointTime</i> use different units of measurement. <i>CheckpointInterval</i> is based on the number of log writes, while <i>MinCheckpointTime</i> specifies the minimum time between consecutive checkpoints.)</p>	50000 log writes	RW
<i>DataDictionaryErrorMaxWait</i>	<p>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</p>	0 (Disabled) Unit: 1 second	RW/Startup

[General]	Description	Factory Value	Access Mode
	should only be enabled when the thread/client mode is used (<i>Srv.ReadThreadMode=2</i>), because the wait blocks the waiting thread.		
<i>DefaultStoreIsMemory</i>	<p>If set to Yes, then new tables are created as in-memory tables if they are created without an explicit STORE clause in the CREATE TABLE statement. If set to No, then by default new tables are stored on disk. You can override the factory value by using the STORE clause in the CREATE TABLE statement.</p> <p>This parameter only applies to products that support solidDB In-memory Engine.</p> <p>Note that system tables are stored on disk, even if this parameter is set to Yes.</p>	No.	RW
<i>DisableIdleMerge</i>	If set to yes, database is set to disable idlemmerge.	No	RW/Startup
<i>FileWriteFlushMode</i>	<p><i>filewriteflushmode=0</i> means no flushing after write or read operations.</p> <p><i>filewriteflushmode=1</i> means flush before reading from the file.</p> <p><i>filewriteflushmode=2</i> means flush after write operations (recommended for vxworks)</p>	0 on most platforms.	RW/Startup
<i>IOThreads</i>	<p>Number of helper I/O threads (per IO device) for read and write purposes.</p> <p>Note: You can restrict the number of write threads with the <i>General.WriterIOThreads</i> parameter. Must be <i>IOThreads > WriterIOThreads</i>. If this rule is violated, the <i>IOThreads</i> parameter takes the precedence (wins).</p>	5	RW/Startup
<i>LockHashSize</i>	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	1000	RW/Startup

[General]	Description	Factory Value	Access Mode
	<p>causes memory overhead. The <i>LockHashSize</i> determines the number of elements in hash table.</p> <p>This information is needed when the server is using pessimistic concurrency control (i.e. locking). The server uses separate arrays for in-memory tables and disk-based tables. This parameter applies to disk-based tables.</p> <p>In general, the more locks you need, the larger this array should be. However, it is difficult to calculate the number of locks that you need, so you will probably need to experiment to find the best value for your applications.</p> <p>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).</p>		
<i>LockWaitTimeOut</i>	<p><i>LockWaitTimeout</i> 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.</p> <p>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.</p> <p>The maximum lock timeout is 1000 seconds. The server won't start if the default lock timeout in <i>solid.ini</i> is more than 1000 seconds.</p> <p>Note: You can set the lock time out for a single connection by using the following SQL command:</p>	<p>30</p> <p>Unit: seconds</p>	RW

[General]	Description	Factory Value	Access Mode
	<p>SET LOCK TIMEOUT timeout_in_se</p> <p>You can change the granularity of the SET LOCK TIMEOUT command from seconds to milliseconds by appending "MS" to the number, e.g.</p> <p>SET LOCK TIMEOUT 500MS</p> <p>Note: The SET LOCK TIMEOUT command does not change the setting in the solid.ini file.</p> <p>See also <i>TableLockWaitTimeOut</i>.</p>		
<i>LongSequentialSearchLimit</i>	Sets the number of sequential fetches after which search is treated as long sequential search	500	RW/Startup
<i>MaxMergeParts</i>	This parameter is used to specify the maximum number of concurrent merge operations, or the number of merge parts.	100	RW/Startup
<i>MaxMergeTasks</i>	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
<i>MaxOpenFiles</i>	Sets the maximum number of files kept concurrently open during solidDB sessions	OS dependent	RW/Startup
<i>MergeInterval</i>	Sets the number of index inserts made in the database that causes the merge process to start	Cache size dependent	RW
<i>MinCheckpointTime</i>	Specifies the minimum time in seconds between two checkpoint operations. SEE ALSO: <i>CheckpointInterval</i> . (Note that <i>CheckpointInterval</i> and <i>MinCheckpointTime</i> use different units of measurement. <i>CheckpointInterval</i> is based on the number of log writes, while <i>MinCheckpointTime</i> specifies the minimum time between consecutive checkpoints.)	0 Unit: 1 second	RW
<i>MinMergeTime</i>	This sets a minimum time (in seconds) between two merge operations. For more information about merge	0	RW

[General]	Description	Factory Value	Access Mode
	operations, see "Solid Bonsai Tree Multiversioning and Concurrency Control" and "Setting the MergeInterval Parameter" in <i>solidDB Administration Guide</i> .		
<i>NetBackupConnect</i>	This sets the connect string to the Netbackup Server.	No factory value.	RW/Startup
<i>NetBackupConnect-Timeout</i>	Sets the maximum time in milliseconds that a netbackup operation waits for a connection to a NetBackup Server.	For example, to set the timeout to 30 seconds use value 30000 (milliseconds). 0 (no timeout)	RW/Startup
<i>NetBackupCopyIni-File</i>	If set to "yes" the <code>solid.ini</code> configuration file is copied to the remote backup directory.	Yes	RW/Startup
<i>NetBackupCopy Log</i>	If set to "yes" the log files are copied to the remote backup directory.	Yes	RW/Startup
<i>NetBackupCopy SolmsgOut</i>	If set to "yes" the <code>solmsg.out</code> message file is copied to the remote backup directory.	Yes	RW/Startup
<i>NetBackupDeleteLog</i>	If set to "yes" the backed-up log files are deleted from the source server after the NetBackup has accomplished.	Yes	RW/Startup
<i>NetBackupDirectory</i>	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
<i>NetBackupRead-Timeout</i>	Sets the maximum time in milliseconds that any operation waits for the response from the NetBackup Server. For example, to set the timeout to 30 seconds use value 30000 (milliseconds).	60 000	RW/Startup
<i>Pessimistic</i>	When you specify PESSIMISTIC concurrency control, the server places locks on rows to control the level of consistency and concurrency when users are submitting queries or updates to rows. The factory value is 'No', that	No	RW/Startup

[General]	Description	Factory Value	Access Mode
	<p>is, the server uses optimistic concurrency control. However, by setting the <i>Pessimistic</i> parameter to 'Yes', you can tell the server to default to pessimistic locking for any new tables that are created AND for any old tables whose concurrency control method was never explicitly set with the ALTER TABLE command.</p> <p>If you set a table's locking mode by using the command</p> <pre>ALTER TABLE base_table_name SET {OPTIMISTIC PESSIMISTIC}</pre> <p>the ALTER TABLE command takes precedence.</p> <p>For a detailed explanation of pessimistic vs. optimistic concurrency control, as well as a discussion of whether the <i>Pessimistic</i> parameter in <i>solid.ini</i> takes precedence over other methods of setting the concurrency control, see solidDB SQL Guide.</p>		
<i>ReadLevelMaxTime</i>	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
<i>Readonly</i>	if set to yes, database is set to read-only mode	No	RW/Startup
<i>SearchBufferLimit</i>	<p>Sets the maximum percentage of search buffers from the total buffered memory reserved for open cursors.</p> <p>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.</p>	50	RW/Startup

[General]	Description	Factory Value	Access Mode
	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.		
<i>StartupForceMerge</i>	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
<i>TableLockWait-Timeout</i>	<p>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.</p> <p>There are times when a transaction will acquire an exclusive lock to a table. If there is a conflict, this setting provides the transaction's wait period until the exclusive or shared lock is released. This parameter is used for synchronized databases only.</p> <p>Table level locks are used when the PESSIMISTIC keyword is explicitly provided in the following solidDB commands:</p> <pre>IMPORT SUBSCRIPTION MESSAGE <i>message_name</i> EXECUTE (only with NO EXECUTE option) MESSAGE <i>message_name</i> FORWARD MESSAGE <i>message_name</i> GET REPLY DROP SUBSCRIPTION</pre> <p>See also <i>LockWaitTimeOut</i>.</p>	30 Unit: 1 second	RW
<i>TransactionEarlyValidate</i>	Transaction early validating is used when this parameter is set to yes. The possible values are yes and no.	Yes	RO/Startup
<i>TransactionHashSize</i>	The hash table contains slots that are occupied by incomplete (i.e. open) transactions. The transaction hash size	4000, but depends	RW/Startup

[General]	Description	Factory Value	Access Mode
	<p>sets the size of the table for open transactions. Once the number of occupied slots increases, the operations with this table become slower.</p> <p>The database offers higher performance when the average number of transactions per slot is lower (5 is a good initial limit for the transaction per slot average).</p> <p>Note that you can monitor the status of this hash table using ADMIN COMMAND 'report <i>filename</i>'. For example:</p> <p>ADMIN COMMAND 'report myfile.txt'</p> <p>The output contains the following related information:</p> <p>tablesize = setting</p> <p>nused = slots taken from hash table</p> <p>list length = sum of all transactions in the table</p>	<p>partly on the cache size. Minimum value is 1000. Maximum is 50,000.</p>	
<i>VersionedPessimisticReadCommitted</i>	<p>If this parameter is enabled, pessimistic D-tables use versioned reads with READ COMMITTED isolation. Read with FOR UPDATE work as before. In other words, pessimistic D-tables work like M-tables.</p>	Yes	RW/Startup
<i>VersionedPessimisticRepeatableRead</i>	<p>If this parameter is enabled, pessimistic D-tables use versioned reads with REPEATABLE READ isolation.</p>	Yes	RW/Startup
<i>WriterIOThreads</i>	<p>Number of helper threads dedicated to writing tasks (per IO device). Must be <i>IOThreads</i> > <i>WriterIOThreads</i>. If this rule is violated, the factory value is used. If <i>IOThreads</i>=1 then the setting <i>WriterIOThreads</i>=0 is enforced.</p>	1	RW/Startup

A.8 HotStandby Section

Table A.5. HotStandby Parameters

<i>[HotStandby]</i>	Description	Factory Value	Access Mode
<i>1SafeMaxDelay</i>	In 1-Safe replication, the maximum delay before a committed transaction is sent to the Secondary (in milliseconds).	1000	RW
<i>2SafeAckPolicy</i>	<p>This specifies the timing of the Secondary's acknowledgement when it receives a transaction from the Primary.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> • 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", i.e. when the Secondary has executed the transaction. • 3 = 2-safe durable. The Secondary server acknowledges when it has made the data durable, i.e. when it has committed the data and written the data to the disk. <p>Not surprisingly, 2-safe durable is the safest approach, and 2-safe received has the fastest response time. However, in practice, the 2-safe received mode provides in most cases sufficient guarantees for data safety hence providing the best compromise between safety and speed.</p> <p>This parameter applies only if the server is using 2-safe replication.</p> <p>NOTE! Although this parameter controls the Secondary server's behavior, this parameter is set on the Primary. The value in the Secondary's <code>solid.ini</code> value is ignored.</p>	1	RW

<i>[HotStandby]</i>	Description	Factory Value	Access Mode
	(See chapters "Determining Whether the Primary's Settings Take Precedence Over the Secondary's" and "Ensuring that Primary and Secondary Parameter Values Are Coordinated" in <i>solidDB High Availability User Guide</i> .)		
<i>AutoPrimaryAlone</i>	If this parameter is set to Yes, then the server is automatically put in PRIMARY ALONE state (rather than PRIMARY UNCERTAIN state) when the connection to the Secondary is broken. If you plan to set this to "yes", please read the very important warnings in "Network Partitions and Dual Primaries" in <i>solidDB High Availability User Guide</i> .	No	RW
<i>CatchupSpeedRate</i>	<p>While the server is performing catchup, it also continues to service database requests from clients. You may use the <i>CatchupSpeedRate</i> parameter to give greater importance to responding to application requests and lower priority to catchup, or vice versa.</p> <p>The speed rate is expressed as a percentage of the maximum available speed dictated by the link and Secondary throughput. Larger numbers mean more emphasis on catchup and less on servicing client requests. Allowed values are 1-99. The factory value is 70.</p>	70	RW
<i>Connect</i>	<p>The <i>Connect</i> parameter indicates the "address" of the other HotStandby server in the pair.</p> <p>The format of the Connect string in the HotStandby section is the same as the format of the <i>Listen</i> parameter in the <i>[Com]</i> section (see <i>solidDB Administration Guide</i> for more details).</p> <p>If you omit this parameter in a server that you intend for HotStandby, then you can set this parameter dynamically by using an ADMIN COMMAND. Until the server has a Connect string, the server can only be in the states that do not involve a connection, i.e. PRIMARY ALONE, SECONDARY ALONE, and STANDALONE.</p>	No factory value.	RW

<i>[HotStandby]</i>	Description	Factory Value	Access Mode
	The <i>Connect</i> parameter is ignored unless the <i>HSBEnabled</i> parameter is set to "yes".		
<i>ConnectTimeout</i>	<p>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.</p> <p>The <i>ConnectTimeout</i> parameter (which is useful only on certain platforms) is only used with certain administration commands. These are:</p> <p>hotstandby connect</p> <p>hotstandby switch primary</p> <p>hotstandby switch secondary</p> <p>For example, to set the timeout to 30 seconds (30000 milliseconds)</p> <p><i>[HotStandby]</i> ConnectTimeout=30000</p> <p>See also <i>PingTimeout</i>.</p>	<p>0 (no timeout)</p> <p>Unit: 1 ms</p>	RW
<i>CopyDirectory</i>	<p>The <i>CopyDirectory</i> parameter in the <i>[HotStandby]</i> section defines a name and location for the HotStandby copy operation that is performed when the user executes the command:</p> <p>ADMIN COMMAND 'hotstandby copy';</p> <p>For example, the parameter may look like:</p> <p><i>[HotStandby]</i> CopyDirectory= C:\Solid\secondary\dbfiles</p>	No factory value	RW

<i>[HotStandby]</i>	Description	Factory Value	Access Mode
	<p>If you provide a relative path for the <i>CopyDirectory</i> parameter, the path will be relative to the directory that holds the Primary server's <i>solid.ini</i> file.</p> <p>This parameter has no factory value, so if the directory is not specified in the <i>solid.ini</i> file, it must be provided in the copy command.</p> <p>Please note that ADMIN COMMAND 'hotstandby netcopy' as the more flexible solution is the recommended way to copy the database.</p>		
<i>HSBEnabled</i>	<p>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.</p> <p>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".</p> <p>To use HotStandby, you must also specify the <i>Connect</i> parameter, either by setting it in the <i>solid.ini</i> file or by using an ADMIN COMMAND to set it.</p>	no	RO
<i>MaxLogSize</i>	Maximum size of the disk-based HSB log. The factory value: unlimited	0 Unit: 1 byte k=KB m=MB	
<i>MaxMemLogSize</i>	When the file-based logging is disabled (<i>Logging.LoggingEnabled=No</i>), 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
<i>NetcopyRpcTimeout</i>	Data transmission acknowledgment timeout for netcopy operation (in milliseconds)	30000 Unit: 1 ms	RW

[HotStandby]	Description	Factory Value	Access Mode
<i>PingInterval</i>	<p>The Primary and Secondary "ping" 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.)</p> <p>The value is equal to the interval (in milliseconds) between two consecutive pings sent by a server. The factory value is 1000 (one second).</p>	<p>1000</p> <p>Unit: 1 ms</p>	RW
<i>PingTimeout</i>	<p>The parameter specifies how long a server should wait before concluding that the other server is down or inaccessible.</p> <p>After the time specified (in milliseconds) has passed the server concludes that a connection is broken and changes the state accordingly. The factory value is 4000 (four seconds).</p> <p>See "PingTimeout and PingInterval Parameters [HotStandby]" in <i>solidDB High Availability User Guide</i>.</p> <p>See also <i>ConnectTimeout</i>.</p>	<p>4000</p> <p>Unit: 1 ms</p>	RW
<i>PrimaryAlone</i>	This parameter is deprecated. Use the <i>AutoPrimary-Alone</i> parameter.	No	RW
<i>SafenessLevel</i>	To set the safeness level of the replication protocol.	<p>Possible values are: 1safe, 2safe and auto</p> <p>By using the "auto" value, you can allow for the safeness level to be dynamically controlled with Solid's standard</p>	RW

<i>[HotStandby]</i>	Description	Factory Value	Access Mode
		SQL durability control statements. If you set <i>SafenessLevel</i> to "auto" and set the durability to relaxed by using the command SET DURABILITY, the safeness level is set to 1-safe, and when you set the durability to strict, the safeness level is set to 2-safe. 2safe	

A.9 IndexFile Section

Table A.6. IndexFile Parameters

<i>[IndexFile]</i>	Description	Factory Value	Access Mode
<i>BlockSize</i>	Sets the block size of the database file in bytes; use multiple of 2 KB: minimum 2 KB, maximum 64 KB	8 KB Unit: 1 byte k=KB	RO

[IndexFile]	Description	Factory Value	Access Mode
<i>CacheSize</i>	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. WARNING: Setting the <i>CacheSize</i> 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 should reduce the <i>CacheSize</i> .	32 MB Unit: 1 byte k=KB m=MB	RW/Startup
<i>ExtendIncrement</i>	Sets the number of blocks of disk space that are allocated at one time when solidDB needs to allocate more space for the database file. Currently, each block is 8KB. E.g. a value of 500 (8KB blocks) corresponds to 4 MB of disk space.	500	RW/Startup
<i>FileSpec_[1... N]</i>	<p>Defines the location and the maximum size of the index file. Note that in solidDB, the term "index file" is used as a synonym for "database file." The parameter accepts the following three arguments: database file name followed with maximum size (in bytes) of the database file, for example:</p> <p>FileSpec_1=c:\solddb\solid.db 200000000</p> <p>This parameter also has an optional argument after the maximum size: device number, which is the physical drive number. The number value itself is not essential, but it is used as a hint for I/O threads, allowing the server to perform database file I/O requests in a parallel manner if you split the file into multiple physical disks. The <i>N</i> in the parameter syntax signifies the number of the file if the database file is divided into multiple files and onto multiple disks. For details, read the section called "FileSpec_[1...N] Parameter".</p>	solid.db 2147483647 (2GB -1)	RW/Startup

[IndexFile]	Description	Factory Value	Access Mode
	<p>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.</p> <p>Note that you may also want to have multiple files on a single disk if your physical disk is partitioned into multiple logical disks and no single logical disk can accommodate the size of the database file you expect to create.</p>		
<i>PreFlushPercent</i>	<p>Sets the percentage of page buffer which is kept clean by the preflush thread.</p> <p>Note that the preflush operations prepare the cache for the allocation of new blocks. The blocks are written onto the disk from the tail of the cache based on a Least Recently Used (LRU) algorithm. Therefore, when the new cache blocks are needed, they can be taken immediately without writing the old contents onto the disk.</p>	1	RW/Startup
<i>ReadAhead</i>	<p>Sets the number of prefetched index reads during long sequential searches.</p> <p>Note that when the I/O manager is handling a long sequential search, it enters a read-ahead operation mode. This mode ensures that the next file blocks of the search in question are read into the cache in advance. This naturally improves the overall performance of sequential searches.</p>	4	RW/Startup
<i>SynchronizedWrite</i>	<p>On UNIX/Linux platforms this parameter may be set to "no" to enact asynchronous I/O. Asynchronous I/O provides, in general, more performance but it can cause higher variance of response latencies (lower latency determinism).</p>	yes	RO

A.10 Logging Section

Table A.7. Logging Parameters

<i>[Logging]</i>	Description	Factory Value	Access Mode
<i>BlockSize</i>	Sets the block size of log files. The log block size may be changed between startups. Logs having block size different than the one set are accepted at recovery. The value has to be a multiplicity of 1 KB. Bigger blocks reduce the overhead of log writing.	16 KB Unit: byte k=KB	RW/Startup
<i>DigitTemplateChar</i>	Specifies the template character that will be replaced in the name template of the log file. See the description of the <i>FileNameTemplate</i> for more details.	#	RW/Startup
<i>DurabilityLevel</i>	<p>This parameter controls whether the transaction durability level is "strict", "relaxed", or "adaptive". If durability is "strict", then writes to the transaction log are synchronous — i.e. as soon as a transaction has been committed, the transaction is written to the transaction log. If durability is "relaxed", then writes are asynchronous — there may be a delay between the time that the transaction is committed and the time that it is logged. For a detailed explanation of "strict" and "relaxed" durability, see Section 6.1, “Logging and Transaction Durability”.</p> <p>The possible values are:</p> <p>1 = relaxed durability</p> <p>2 = adaptive durability. This value applies only to HSB (HotStandby) Primary servers.</p> <p>3 = strict durability</p> <p>The server's durability level may be set dynamically by using the command:</p> <p>ADMIN COMMAND 'parameter Logging.DurabilityLevel=n';</p>	2	RW

[Logging]	Description	Factory Value	Access Mode
	<p>where <i>n</i> is one of the valid values for this parameter.</p> <p>Each connection may override this <code>solid.ini</code> parameter by using the SET DURABILITY or SET TRANSACTION DURABILITY command. See chapter "SET" in <i>solidDB SQL Guide</i>.</p> <p>Note that the <code>DurabilityLevel</code> parameter affects the server's behavior only if transaction logging is turned on. If you turn off transaction logging by setting</p> <pre>[Logging] LogEnabled=No</pre> <p>then your data will not be logged to disk, regardless of the setting of <code>DurabilityLevel</code>. If <code>LogEnabled</code> is set to No and <code>DurabilityLevel</code> is set, then the server will briefly display a warning message at the time that it starts.</p> <p><code>DurabilityLevel</code> is not the only configuration parameter that influences how the server writes information to logs. You may also want to read about the <code>LogWriteMode</code> parameter, which also offers some options that allow you to trade off speed and reliability. If you are using the Solid CarrierGrade option (HotStandby), you may also want to read about the <code>2SafeAckPolicy</code> parameter.</p>		
<i>FileFlush</i>	<p>This parameter controls log file flush behavior. This parameter is only valid for platforms supporting Synchronized I/O Data Integrity Completion. These are such as Solaris, HP-UX, and Linux.</p> <p>When set to no in these platforms, the operating system, rather than the solidDB engine, flushes the log file.</p>	yes	RW/Startup

[Logging]	Description	Factory Value	Access Mode
<i>FileNameTemplate</i>	<p>Defines the path and naming convention used when creating log files. These log files contain information used to recover data if the server crashes.</p> <p>To be more specific, this parameter defines at least the naming convention used when creating log files, but not necessarily the path. If this is the case, the <i>Logging.LogDir</i> parameter defines the path. For more information, see the <i>LogDir</i> parameter description.</p> <p>Template characters (e.g. "#") are replaced with sequential numbers; for example, the following file entry instructs solidDB to create log files in directory C:\solid\log and to name them sequentially starting from sol00001.log.</p> <pre>FileNameTemplate = c:\solid\log\sol#####.log</pre> <p>Your template may use between 4 and 10 template characters. If you do not want to use the "#" sign as a template character, you may specify a different character by setting the parameter <i>DigitTemplateChar</i>.</p> <p>If the number of log files would exceed the maximum possible number (e.g. all names from sol00001.log to sol99999.log are used up), then the server will give an error message and exit. The error message will be similar to the following:</p> <pre>"Error: Illegal log file name template. Most likely the log file name template specified in solid.ini ... contains too few or too many sequence number digit positions. There should be at least 4 and at most 10 digit</pre>	sol#####.log	RW/Startup

[Logging]	Description	Factory Value	Access Mode
	<p>positions."</p> <p>To achieve better performance, the log files must be stored to a local drive using local disk names to avoid problems with network I/O.</p>		
<i>LogDir</i>	This parameter sets the directory prefix of the log file path specified by using the Logging. <i>FileNameTemplate</i> parameter. Effectively, it specifies the log file directory if <i>FileNameTemplate</i> only specifies the file name (default). The default value is the server working directory. The specified directory has to exist prior to starting the server.	"." (the server's working directory)	RW/Startup
<i>LogEnabled</i>	Specifies whether transaction logging is enabled or not. If transaction logging is disabled, you will get better performance but lower transaction durability (if the database engine shuts down unexpectedly, then you lose any transactions since the last checkpoint). Note that this parameter applies to in-memory tables as well as disk-based tables.	yes	RW/Startup
<i>LogWriteMode</i>	<p>Specifies the mode in which the log will be written. The following two modes are available:</p> <ul style="list-style-type: none"> 0: ping-pong method 2: Overwrite method (factory value) <p>The choice of logging method depends on the log file media and the level of security required. For details on each of these methods, read Section 3.10.10, "Transaction Logging".</p>	2 (Overwrite method)	RW/Startup
<i>MinSplitSize</i>	When this file size is reached, logging will be continued to the following log file after the next checkpoint	<p>10 MB</p> <p>Unit: 1 KB k=KB m=MB</p>	RW/Startup

[Logging]	Description	Factory Value	Access Mode
<i>RelaxedMaxDelay</i>	This sets the maximum time in milliseconds that the server waits until the committed transaction(s) are written to the log. This parameter applies only when the transaction durability level is set to RELAXED (with the <i>DurabilityLevel</i> parameter or the SET DURABILITY statement). The units are milliseconds. Minimum allowed value: 100 (i.e. 100 milliseconds).	5000 milliseconds (5 seconds) Unit: 1 ms	RW/Startup
<i>SyncWrite</i>	This parameter applies only to platforms, such as Solaris, HP-UX, and Linux, which support Synchronized I/O Data Integrity Completion. When set to yes, solidDB assumes that the platform supports Synchronized I/O Data Integrity Completion. It should be set to No on all other platforms.	no	RW/Startup

A.11 MME Section





Note



The *DefaultStoreIsMemory* parameter (in the *[General]* section of the *solid.ini* file) is also related to solidDB In-memory Engine capability. For more information, see Section A.7, “General Section”.

Table A.8. MME parameters

[MME]	Description	Factory Value	Access Mode
<i>ImdbMemoryLimit</i>	This sets an upper limit on the amount of memory (virtual memory) that the server will allocate for in-memory tables and indexes on in-memory tables. Note that "in-memory tables" includes Temporary Tables and Transient Tables, as well as "normal" (persistent) in-memory tables. The limit may be specified in bytes, kilobytes (kb), megabytes (mb), or gigabytes (gb). For example: <i>ImdbMemoryLimit</i> =1073741824	0 Unit: 1 byte k=KB m=MB g=GB	RW

[MME]	Description	Factory Value	Access Mode
	<p>ImdbMemoryLimit=1048576kb ImdbMemoryLimit=1024MB ImdbMemoryLimit=1GB</p> <p>If you use the value 0, it means "no limit".</p> <p>As a general rule, for servers with 1GB or less of memory, the maximum amount that you should allocate to in-memory tables is usually 30% - 70% of the system's physical memory. The more memory the system has, the larger the percentage of it you may use for in-memory tables.</p> <p>For more details about controlling memory usage of in-memory tables, see <i>solidDB In-Memory Database User Guide</i>.</p> <p>Note: This parameter only applies only to solidDB In-memory Engine tables. It does not apply to other versions of solidDB or to disk-based tables.</p> <p>You can change this with the command:</p> <pre>ADMIN COMMAND 'parameter MME.ImdbMemoryLimit=n[kb mb gb]';</pre> <p>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 <code>solid.ini</code> file at shutdown.</p> <p> Caution</p> <p>We strongly recommend that you ensure that your in-memory tables will fit within the available physical memory. If you exceed the amount of physical memory available, performance will de-</p>		

[MME]	Description	Factory Value	Access Mode
	crease significantly. If you use up all of the available virtual memory, the server will abruptly limit inserts, updates, etc. and will return error codes.		
<i>ImdbMemoryLowPercentage</i>	<p>Once you have set <i>ImdbMemoryLimit</i>, you may set this additional parameter to give you advance warning before you use up all of memory. This <i>ImdbMemoryLowPercentage</i> parameter allows you to indicate what percentage of memory you may use before the server starts limiting your ability to insert rows into in-memory tables, etc. For example, if <i>ImdbMemoryLimit</i> is 1000MB and <i>ImdbMemoryLowPercentage</i> is 90 (percent), then the server will stop accepting inserts when you've used up 900 megabytes of memory for your in-memory tables.</p> <p>Valid values are between 60 and 99 (percent).</p> <p>For more details about controlling memory usage of in-memory tables, see <i>solidDB In-Memory Database User Guide</i>.</p> <p> Note</p> <p>This parameter only applies to solidDB In-memory Engine tables. It does not apply to other versions of solidDB or to disk-based tables.</p>	90	RW/Startup
<i>LockEscalationEnabled</i>	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 <i>LockEscalationEnabled</i> is set to yes, the lock level is escalated from row-level to table-level after a specified number of	yes	RW/Startup

[MME]	Description	Factory Value	Access Mode
	<p>rows (in the same table) have been locked within the current transaction. Lock escalation improves performance, but reduces concurrency, because it means that other users are temporarily unable to use the same table, even if they want to use different rows within that table. See the parameter <i>LockEscalationLimit</i>.</p> <p>The value may be "yes" or "no".</p> <p> Note</p> <p>This parameter applies to in-memory tables only.</p>		
<i>LockEscalationLimit</i>	<p>If <i>LockEscalationEnabled</i> is set to yes, then this parameter indicates how many rows must be locked (within a single table) before the server will escalate lock level from row-level to table-level. (See <i>LockEscalationEnabled</i> for more details.)</p> <p>The value may be any number from 1 to 2,147,483,647 ($2^{32}-1$).</p> <p> Note</p> <p>This parameter applies to in-memory tables only.</p>	1000	RW/Startup
<i>LockHashSize</i>	<p>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 <i>LockHashSize</i> determines the number of elements in hash table.</p> <p>This information is needed when the server is using pessimistic concurrency control (i.e. locking). The server uses separate arrays for in-memory tables and disk-based tables. This parameter applies to in-memory tables.</p>	1000000	RW/Startup

[MME]	Description	Factory Value	Access Mode
	<p>In general, the more locks you need, the larger this array should be. However, it is difficult to calculate the number of locks that you need, so you will probably need to experiment to find the best value for your applications.</p> <p>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).</p>		
<i>MaxCacheUsage</i>	<p>The value of <i>MaxCacheUsage</i> 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 <i>MaxCacheUsage</i> at most half of the D-table cache (<i>IndexFile.CacheSize</i>) is used for checkpointing M-tables. Value <i>MaxCacheUsage=0</i> sets the value unlimited, which means that the cache usage is <i>IndexFile.CacheSize/2</i>.</p>	8MB	RW/Startup
<i>ReleaseMemoryAtShutdown</i>	<p>When set to "yes", this tells the server that when it shuts down it should explicitly release memory used by in-memory tables, rather than relying on the operating system to clean up all memory associated with this process. Some operating systems (like VxWorks) may require you to set this to "yes" to ensure that all memory is released.</p> <p>The possible values are yes and no.</p> <p>The factory value is no because shutting down the server is faster that way.</p>	No	RW/Startup

A.12 Sorter Section

Table A.9. Sorter Parameters

<i>[Sorter]</i>	Description	Factory Value	Access Mode
<i>BlockSize</i>	Block size of the external sorter files. With the factory value 0, the database block size is used.	0	RW/Startup
<i>MaxCacheUsePercent</i>	<p>This parameter sets the maximum percentage of cache pages that can be used for sorting. The valid values range from 10% to 50%. E.g. if the <i>CacheSize</i> (in the <i>IndexFile</i> section of the <i>solid.ini</i> file) is 20MB, and if <i>MaxCacheUsePercent</i> is 25, then a maximum of 5MB of memory is available for sorting.</p> <p>If you specify both the <i>MaxCacheUsePercent</i> and the <i>MaxMemPerSort</i>, the values must be compatible. You get an error message if the following is not true: $MaxCacheUsePercent \times CacheSize \geq MaxMemPerSort$</p>	25 (that is, 25 percent)	RW/Startup
<i>MaxFilesTotal</i>	Maximum number of files used for sorting	100	RW/Startup
<i>MaxMemPerSort</i>	This parameter sets the maximum memory available in bytes for one sort (that is, sorting the result set of one query). This value must not exceed the amount of memory available to the sorter (see <i>MaxCacheUsePercent</i> for more information).		RW/Startup
<i>SorterEnabled</i>	This parameter enables or disables the usage of the external sorter.	Yes	RW/Startup
<i>TmpDir_[1... N]</i>	When this parameter is specified in the configuration file, the external sorter algorithm is enabled. The external sorter algorithm is used for sorting processes that do not fit in main memory. The parameter defines the name of the directory (or directories) that contain temporary files created when using the external sorter algorithm. The <i>N</i> signifies the file directory number if more than one directory is used to store the temporary file. For example:	Defaults to ".", in other words, the current directory (the directory from which the server was started).	RW/Startup

[Sorter]	Description	Factory Value	Access Mode
	TmpDir_1=c:\solddb\temp1 TmpDir_2=d:\solddb\temp2		

A.13 SQL Section

Table A.10. SQL Parameters

[SQL]	Description	Factory Value	Access Mode
<i>AllowDuplicateIndex</i>	If set to yes, allows duplicate index definitions. This is a backward compatibility parameter. In versions preceding 4.5, it was possible to create duplicate indexes.	no	RO
<i>CharPadding</i>	<p>When set to yes, enforces SQL standard padding of CHAR values with blanks (right-filled) to the length defined for the column. With the default setting (no), the blanks are discarded. The value of the parameter does not affect comparisons (where blanks are always discarded).</p> <p>This feature is not implemented in the Solid SQL Editor (solsql). Use ODBC3 or JDBC2 drivers with this feature. Notice also that this parameter affects the ODBC and JDBC driver behaviour.</p>	no	RO
<i>ConvertOr-sToUnionsCount</i>	This parameter specifies the maximum number of OR operations that may be converted to UNION operations. Note that this parameter does not force the optimizer to convert OR operations to UNION operations; it merely sets a maximum limit on the number of OR operations that the server may convert to UNION operations.	0	RW/Startup
<i>CursorCloseAt-TransEnd</i>	By default, the Solid ODBC driver closes all the cursors opened from the user connection when a commit is called with <code>SqlTransact</code> from this connection. If this parameter is set to No, the cursors are kept open.	yes	RO

[SQL]	Description	Factory Value	Access Mode
<i>EmulateOldTimestamp-Diff</i>	If included in the <code>solid.ini</code> file and set to "Yes", the old <code>TIMESTAMPDIFF</code> behavior is emulated by the server. This old behavior returns the integer number of intervals of type <code>interval</code> by which <code>timestamp_exp2</code> is greater than <code>timestamp_exp1</code> . Otherwise, the default is the new behavior which returns the integer number of interval as the amount of full units between <code>timestamp_exp1</code> and <code>timestamp_exp2</code> .	"No"	RW/Startup
<i>EnableHints</i>	<p>If this parameter is included in the <code>solid.ini</code> file and set to "Yes", hints are enabled. If set to "No," hints are disabled.</p> <p>For details on hints, read "Using Optimizer Hints" in <code>solidDB SQL Guide</code>.</p> <p>Sometimes hints in queries may produce undesirable effects. They may be disabled by setting this parameter to "no"</p>	yes	RW/Startup
<i>Info</i>	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 <i>InfoFileName</i> , or into <code>soltrace.out</code> if <i>InfoFileName</i> is not defined.	0	RW/Startup
<i>SQLInfo</i>	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 <i>InfoFileName</i> , or into <code>soltrace.out</code> if <i>InfoFileName</i> is not defined.	0	RW/Startup
<i>InfoFileFlush</i>	If set to yes, flushes info file after every write operation	yes	RW/Startup
<i>InfoFileName</i>	Default info file name. The default name is <code>soltrace.out</code> . Since the <code>soltrace.out</code> file may contain information from several sources, we recommend that you explicitly set <i>InfoFileName</i> to another name if you set the <i>Info</i> or <i>SQLInfo</i> parameters to a number larger than 0.	<code>soltrace.out</code>	RW/Startup
<i>InfoFileSize</i>	Sets the maximum size of the info file.	1 MB	RW/Startup

[SQL]	Description	Factory Value	Access Mode
<i>IsolationLevel</i>	<p>Possible values:</p> <p>3 (SERIALIZABLE)</p> <p>2 (REPEATABLE READ)</p> <p>1 (READ COMMITTED)</p> <p>This is the default transaction isolation level. For more information about transaction isolation levels, see the description of the SET TRANSACTION ISOLATION command (part of solidDB SQL Guide, Appendix B, Solid SQL Syntax), and chapter Choosing Transaction Isolation Levels in <i>solidDB Administration Guide</i>.</p> <p>In addition to setting this parameter in the <code>solid.ini</code> file, you may also set the value by executing the following command:</p> <pre>ADMIN COMMAND 'parameter SQL.IsolationLevel={1 2 3}'</pre> <p>Note that if you execute this as an admin command, then it takes effect after the server is restarted.</p> <p>Note that in version 4.0 and later, in-memory tables will not work with <i>IsolationLevel</i> set to SERIALIZABLE.</p>	2 (Repeatable Read)	RW/Startup
<i>MaxBlobExpressionSize</i>	<p>Certain string operations use only the first N bytes of a character value, not the entire value. For example, the LOCATE() operation checks only the first N bytes of the string. If you want to tell the server to check further into (or less far into) long strings, you may set this parameter. By default, the units are kilobytes — e.g. "64" means 64KB You may specify "MB" if you want to express the units in megabytes. This parameter applies to all the character data types, including CHAR, VARCHAR, LONG VARCHAR, WCHAR,</p>	<p>1024KB (1MB)</p> <p>Unit: 1 KB m=MB</p>	RW/Startup

[SQL]	Description	Factory Value	Access Mode
	WVARCHAR, and LONG WVARCHAR. Since the Wide character data types use 2 bytes per character, the number of characters searched is half the number of bytes. E.g. if you set <i>MaxBlobExpressionSize</i> to 64K bytes, then the first 32K characters of Wide character data types will be searched.		
<i>MaxNestedProcedures</i>	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
<i>MaxNestedTriggers</i>	Sets the maximum number of allowed nested triggers. This maximum number includes both direct and indirect nesting, so both $A \rightarrow A \rightarrow A$ and $A \rightarrow B \rightarrow A$ are counted as three nested triggers.	16	RW/Startup
<i>ProcedureCache</i>	Specifies the number of procedures which set the size of cache memory for parsed procedures.	10	RW/Startup
<i>SimpleOptimizer-Rulest</i>	Instead of using full optimization rules, simplified one can be used by setting the value to "yes".	No	RW/Startup
<i>SortArraySize</i>	The size of the array that SQL uses when ordering the result set of a query. The units are "rows" — e.g. if you specify a value of 1000, then the server will create an array big enough to sort 1000 rows of data.	2000	RW/Startup
<i>TimestampDisplay-Size19</i>	If this parameter is included in the <code>solid.ini</code> file and set to "Yes", it sets the precision (i.e. maximum number of digits) of data type timestamp to 19. In this case, the timestamp is presented as <i>yyyy-mm-dd hh:mm:ss</i> .	No	Startup
<i>TriggerCache</i>	Specifies the number of triggers which set the size of cache memory that each user has for triggers.	20	RW/Startup

A.14 Srv Section

Table A.11. Srv Parameters

<i>[Srv]</i>	Description	Factory Value	Access Mode
<i>AbortTimeout</i>	Specifies the time in minutes after an idle transaction is aborted; negative or zero value means infinite.	120 Unit: 1 min	RW/Startup
<i>AdaptiveRowsPerMessage</i>	This parameter takes the average number of rows returned to the client as the rows per message value. Of course, the start value grows as more rows are fetched. If set to no, the <i>RowsPerMessage</i> parameter value is used. That is also the default value.	yes	RW/Startup
<i>AllowConnect</i>	If set to no, only connections from Remote Control, Solid SQL Editor, or SolidConsole are allowed	yes	RW/Startup
<i>At</i>	<p>The syntax is:</p> <pre>At = At_string At_string ::= timed_command [,timed_command] timed_command ::= [day] HH:MM argument day ::= sun / mon / tue / wed / thu / fri / sat</pre> <p>If entered, allows you to specify a command to automate an administrative task, such as executing system commands, creating backups, checkpoints, and database status reports. For example:</p>	(no factory value)	RW

[Srv]	Description	Factory Value	Access Mode
	<p>At = 20:30 makecp, 21:00 backup, sun 23:00 shutdown</p> <p>If you specify a backup, the default backup directory is the one set with the <i>BackupDirectory</i> parameter in the <i>General</i> section.</p> <p>If the day is not given, the command is executed daily.</p> <p>There is no factory value for this parameter.</p>		
<i>ConnectionCheckInterval</i>	When the <i>ReadThreadMode</i> parameter is set to 2 (default), the server doesn't detect a broken connection until it tries to write something back to the client. This parameter specifies the number of seconds between connection status checks in thread/client mode.	10 Unit: seconds	RW/Startup
<i>ConnectTimeOut</i>	Specifies the continuous idle time in minutes after a connection is dropped; negative or zero value means infinite.	480 Unit: 1 min	RW/Startup
<i>DatabaseSizeReportInterval</i>	<p>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.</p> <p>This parameter is useful, for example, when tracing unexpected database size growth.</p> <p>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.</p>	0 MB	RO/Startup
<i>DisableOutput</i>	Disables generation of the solmsg.out and the solerror.out files. For details on these files, read Section 3.7, "Viewing the solidDB Message Log". To disable file generation, this parameter must be included in the solid.ini file and set to yes. If this parameter	no	RW/Startup

[Srv]	Description	Factory Value	Access Mode
	is set to no or not included in the <code>solid.ini</code> file, the log files are generated.		
<i>Echo</i>	If set to yes, contents of <code>solmsg.out</code> file are displayed also at the server's command window.	no	RW/Startup
<i>ExecRowsPerMessage</i>	<p>This parameter specifies how many rows are returned in one network message from the server to the client when the client uses the <code>SQLExecute</code> call with <code>SELECT</code> statements. If your <code>SELECT</code> statements usually return a large number of rows, setting this to an appropriate value can improve performance significantly.</p> <p>See also the <i>RowsPerMessage</i> configuration parameter.</p>	2	RW/Startup
<i>ForceThreadsToSystemScope</i>	<p>This parameter applies only to symmetric multi-process (SMP) Solaris operating systems, in which the default scope provided by the threads of the runtime library can be set to process scope, system scope, or light weight process (lwp) scope. (In Sun's terminology, "threads" are "lightweight processes".)</p> <p>A yes value may significantly improve the server's performance in a multi-CPU machine. (The actual performance improvement depends on how evenly the workload is already spread across your CPUs.) A no value usually provides slightly better performance in single-CPU systems.</p> <p>To fully understand how this parameter works, you must understand the threading facilities of Solaris. An explanation of the Solaris threading facilities is beyond the scope of this manual. However, it may be helpful to understand that when this parameter is set to yes, it forces lwp threads to be run in system scope, instead of process scope. A Yes setting allows Solaris to schedule solidDB threads on any available CPU. This reduces bottlenecks and enhances the parallelization of operations, including I/O.</p>	Servers compiled for Solaris default to Yes. All other servers default to no.	RW/Startup

[Srv]	Description	Factory Value	Access Mode
<i>LocalStartTasks</i>	<p>Number of server's internal tasks (see footnote 1) that execute the local background statements that were started with command START AFTER COMMIT (without FOR EACH REPLICA).</p> <p>Valid values range from 1 - 100.</p>	1	RW/Startup
<i>MaxBgTaskInterval</i>	<p>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".</p> <p>The units are seconds.</p> <p>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 <i>MaxBgTaskInterval</i> 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.</p> <p>You may want to set or adjust this parameter if you get errors similar to the following:</p> <p>Error 08004: [Solid][SOLID ODBC Driver]</p> <p>[SOLID]SOLID Server Error 14507: Maximum number of licensed user connections exceeded</p>	2 (seconds)	RW/Startup

[Srv]	Description	Factory Value	Access Mode
	<p>This parameter only applies to the server's own internal administrative tasks. It does not affect the scheduling of user tasks.</p> <p>WARNING: <i>MaxBgTaskInterval</i> 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.</p> <p>Setting <i>MaxBgTaskInterval</i> to a small enough value may reduce performance and may reallocate some time from high-priority tasks to low-priority tasks. This is particularly likely to happen in "real-world" situations because the customers who use this parameter are most likely to be the customers with busy systems (that is, systems that were so busy they did not check low-priority connections often enough to notice that they had been disconnected). However, because the parameter only affects server administrative tasks, not user tasks, the effect is generally small.</p>		
<i>MaxConstraintLength</i>	<p>This parameter controls the maximum number of bytes that the server will search through in a string, for example in WHERE clauses such as:</p> <pre>WHERE LOCATE(sought_string, column1)</pre> <p>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).</p> <p>This parameter applies to strings that have the following data types:</p> <p>CHAR(#)</p> <p>VARCHAR(#)</p>	254 (254 bytes = 254 ASCII characters, or 127 Unicode characters)	RW

[Srv]	Description	Factory Value	Access Mode
	<p>It does not apply to strings that have the data type(s):</p> <p>LONG VARCHAR</p> <p>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.</p>		
<i>MaxOpenCursors</i>	The maximum number of cursors that a database client can have simultaneously open.	1000	RW/Startup
<i>MaxRPCDataLen</i>	This allows users to specify the maximum string length of a single SQL statement sent to the server. This is particularly useful if you are sending CREATE PROCEDURE commands that are longer than 64K. The value should be between 64K (65536) and 1024K (1048576). If the value is less than 64K, the server will use a minimum of 64K.	512K (524288)	RW/Startup
<i>MaxStartStatements</i>	Maximum number of simultaneous "uncommitted" START AFTER COMMIT statements. Valid values range from 0 - 1000000.	10000	RW/Startup
<i>MemoryReportLimit</i>	This parameter defines the minimum size for memory allocations after which reporting to solmsg.out is done.	100 MB	RW/Startup
<i>MemoryReportDelta</i>	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
<i>MemorySizeReportInterval</i>	<p>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.</p> <p>This parameter is useful, for example, when tracing unexpected memory growth in the server.</p>	0 MB	RO/Startup

[Srv]	Description	Factory Value	Access Mode
	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.		
<i>MessageLogSize</i>	The maximum size of the <code>solmsg.out</code> file in bytes.	1 MB Unit: 1 byte k=KB m=MB	RW/Startup
<i>Name</i>	Specifies the informal name of the server, equivalent to the <code>-n</code> command line option.		RW/Startup
<i>NetBackupRootDir</i>	Sets the root directory for the network backups in NetBackup Server. The path is relative to the working directory.	The working directory	RW
<i>PessimisticTableUseNFetch</i>	<p>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.</p> <p>If pessimistic tables are used, they force the <i>RowsPerMessage</i> value to 1 if the query locks any rows. You can enable the <i>RowsPerMessage</i> for pessimistic tables by enabling the <i>PessimisticTableUseNFetch</i> parameter. By default, it is disabled.</p>	No	RW/Startup
<i>PrintMsgCode</i>	Causes a unique 8-character message code to be inserted before each status and error message in the message log files (<code>solmsg.out</code> and <code>solerr.out</code>).	no	RW/Startup
<i>ReadThreadMode</i>	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 <i>Srv.Threads</i> . 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.	2	RW/Startup

[Srv]	Description	Factory Value	Access Mode
	<p>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.</p> <p>Some operating systems may limit the maximum number of threads allowed, and setting this parameter's value to 2 may cause the server to request more threads than the OS allows. If you try to exceed the number of threads allowed, you will get a message similar to the following:</p> <pre>"Failed to create thread 'dnet_clientthread' ".</pre> <p>(msgcode 30146)</p>		
<i>RemoteStartTasks</i>	Number of Replica server's internal tasks (see footnote 1) inside the server that execute the remote background statements started at Master with command START AFTER COMMIT... FOR EACH REPLICA. Valid values range from 1 - 100.	1	RW/Startup
<i>RowsPerMessage</i>	<p>Specifies the number of rows returned from the server in one network message when an <code>SQLFetch</code> call is executed.</p> <p>See also the <i>ExecRowsPerMessage</i> configuration parameter.</p>	100	RW/Startup
<i>Silent</i>	If set to yes, no output is generated to the server's command window. Only license information is displayed.	No	RW/Startup
<i>Statement-MemoryTraceLimit</i>	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 <code>solmsg.out</code> file.	0 MB	RO/Startup
<i>Threads</i>	If the <i>Srv.ReadThreadMode</i> parameter is set to 0, this parameter specifies the number of concurrent threads that the server uses to process user requests. The helper	5	RW/Startup

[Srv]	Description	Factory Value	Access Mode
	threads, such as I/O threads, are not included in the count. If the value of <i>Srv.ReadThreadMode</i> is other than 0, the value of this parameter is insignificant, as the server controls the number of threads automatically.		
<i>TraceLogSize</i>	<p>This parameter allows you to limit the maximum size of the trace log file. The size is specified in bytes; for example, <i>TraceLogSize=10000</i> limits the size of the trace log file to 10000 bytes. The trace log file is the file to which the server writes information when you turn on monitoring. (For information about turning on monitoring, see the description of ADMIN COMMAND 'monitor...' in Appendix B, "solid SQL Syntax", in <i>solidDB SQL Guide</i>, and the "-m" command-line option in Appendix C, <i>solidDB Command Line Options</i>.)</p> <p>Monitoring uses the file named <code>soltrace.out</code> for output. Note that after reaching this maximum size, the server will:</p> <ol style="list-style-type: none"> 1) delete any existing file named <code>soltrace.bak</code>; 2) rename the current <code>soltrace.out</code> file to <code>soltrace.bak</code>; and 3) start a new <code>soltrace.out</code> file. 	<p>1 megabyte</p> <p>Unit: 1 byte k=KB m=MB</p>	RW/Startup
<i>TraceSecDecimals</i>	Number of second decimals in trace outputs. Allowed values are from 0 to 3.	0	RW/Startup

FOOTNOTE 1: In this context, "task" means solidDB's internal task. Do not confuse this with "thread" or with the term "task" as it is used in some Real-Time Operating Systems such as Wind River Systems VxWorks. A task is just an operation that has to be executed, such as checkpoint, backup, or SQL statement. In this case, we can have 1 to N tasks that execute the background operations. More tasks mean that background tasks reserve more resources and are handled faster — and that other operations (e.g. interactive ones) will get fewer resources and be handled more slowly.

A.15 Synchronizer Section

Table A.12. Synchronizer Parameters

<i>[Synchronizer]</i>	Description	Factory Value	Access Mode
<i>ConnectStrForMaster</i>	<p>This parameter indicates the connection string that the master must use to communicate with the replica. This information is read when the server is started, and sent to the master as part of each message from the replica to the master.</p> <p>For example,</p> <pre>ConnectStrForMaster= tcp replicahost 1316</pre>	none	RW/Startup
<i>MasterStatement-Cache</i>	The size of the statement cache used during one propagation in Master. The statement cache is used to store prepared statements received by Master in one propagation from Replica.	10	RW/Startup
<i>RpcEventThreshold-ByteCount</i>	<p>This parameter controls how frequently the server posts events to indicate how many bytes have been sent or received in the current synchronization message. The units are measured in bytes; the smaller the value (that is, the smaller the number of bytes), the less frequently events are posted. Note that you cannot use suffixes such as "K" or "M" to indicate Kilobytes or Megabytes.</p> <p>The factory value is 0, which means that no events are posted.</p> <p>For more information, see the solidDB SmartFlow Data Replication Guide.</p>	0	RW/Startup
<i>RefreshIsolation-Level</i>	With this parameter, you can select the transaction isolation level for refresh operations instead of using the <code>solid.ini</code> default value. The possible values are	Defaults to <i>SQL.Isolation-Level</i>	RW/Startup

[Synchronizer]	Description	Factory Value	Access Mode
	1. READ COMMITTED 2. REPEATABLE READ		
<i>RefreshReadLevel-Rows</i>	With this parameter, you can define the number of rows after the read level is released in the master if the used isolation level is READ COMMITTED. In other cases, the read level is kept for the full time of the refresh operation. The read level denotes a snapshot-consistent version of the data in the whole database. By releasing the read level, you avoid keeping too much data in main memory during the refresh operation.	1000	RW
Note:	The <i>RemoteStartTasks</i> parameter in the <i>Srv</i> section is also related to SmartFlow/Synchronization.		RW/Startup
<i>ReplicaRefreshLoad</i>	This parameter defines the amount of system processing capacity (as percentage) used to perform a refresh in Replica. By default, the full power is used. If some capacity is to be secured for local processing, in parallel with refresh, a lower value may be set.	100	RW

Appendix B. Client-Side Configuration Parameters

The client-side configuration parameters are stored in the `solid.ini` configuration file and are read when the client starts.

Generally, the factory value settings offer the best performance and operability, but in some special cases modifying a parameter will improve performance. You can change the parameters by editing the configuration file `solid.ini`.

The parameter values set in the client side configuration file come to effect each time an application issues a call to the `SqlConnect` ODBC function. If the values are changed in the file during the program's run time, they affect the connections established thereafter.

B.1 Setting Client-Side Parameters through the `solid.ini` Configuration File

When the `solidDB` is started, it attempts to open the configuration file `solid.ini`. If the file does not exist, `solidDB` will use the factory values for the parameters. If the file exists, but a value for a particular parameter is not set in the `solid.ini` file, `solidDB` will use a factory value for that parameter. The factory values may depend on the operating system you are using.

By default, the client looks for the `solid.ini` file in the current working directory, which is normally the directory from which you started the client. When searching for the file, the `solidDB` uses the following precedence (from high to low):

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

B.1.1 Rules for Formatting the Client-Side `solid.ini` File

When you format the client-side `solid.ini` file, the same rules apply as for the server-side `solid.ini` file. For more information, refer to section *Rules for Formatting the `solid.ini` File* in *solidDB Administration Guide*.

Example B.1. Client-Side `solid.ini` File

```
[Com]
;use this connect string of no data source given
Listen = tcp host1.acme.com 1315

[Client]
;at SQLConnect, timeout after this time (ms)
ConnectTimeout = 5000

;at any ODBC network request, timeout after this time (ms)
ClientReadTimeout = 10000

[DataSources]
Primary_Server = tcp irix1 1315, The Primary Server
Secondary_Server = tcp irix2 1315, The Secondary Server
```

B.2 Descriptions of Client-Side Configuration Parameters

There is one table below for each section of the `solid.ini` file. The sections (and tables) are:

- Com
- Data Sources
- Client

B.3 Communication Section

Table B.1. Communication Parameters

<i>[Com]</i>	Description	Factory Value
<i>ClientReadTimeout</i>	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	60 000

[Com]	Description	Factory Value
	the connect string option "-r" and, further on, with the ODBC attribute <i>SQL_ATTR_CONNECTION_TIMEOUT</i> . Note: applies for the TCP protocol only.	
<i>Connect</i>	The <i>Connect</i> parameter defines the default network name (connect string) for a client to connect to when it establishes a connection to a server. This value is used when the <i>SQLConnect</i> () call is issued with an empty data source name.	tcp localhost 1964
<i>ConnectTimeout</i>	The <i>ConnectTimeout</i> 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 <i>SQL_ATTR_LOGIN_TIMEOUT</i> . Note: applies for the TCP protocol only.	OS-specific
<i>Trace</i>	If this parameter is set to yes, trace information on network messages for the established network connection is written to a file specified with the <i>TraceFile</i> parameter. The factory value for the <i>TraceFile</i> parameter is <i>soltrace.out</i> .	no
<i>TraceFile</i>	If the <i>Trace</i> parameter is set to yes, trace information on network messages is written to a file specified with this <i>TraceFile</i> parameter.	<i>soltrace.out</i> (written to the current working directory of the server or client depending on which end the tracing is started)

B.4 Data Sources

Table B.2. Data Source Parameters

[Data Sources]	Description	Factory Value	Access Mode
<i>logical name = network name, Description</i>	These parameters can be used to give a logical name to a solidDB server in a <i>solid.ini</i> file of the client application		N/A

[Data Sources]	Description	Factory Value	Access Mode
	plication. For details, read section <i>Logical Data Source Names</i> in <i>solidDB Administration Guide</i> .		

B.5 Client

Table B.3. Client Parameters

[Client]	Description	Factory Value
<i>RowsPerMessage</i>	This parameter specifies the number of rows returned from the server in one network message when an <code>SQLFetch</code> call is executed. See also the <i>ExecRowsPerMessage</i> configuration parameter.	decided by the server
<i>ExecRowsPerMessage</i>	This parameter specifies how many rows are returned in one network message from the server to the client when the client uses the <code>SQLExecute</code> call with <code>SELECT</code> statements. If your <code>SELECT</code> statements usually return a large number of rows, setting this to an appropriate value can improve performance significantly. See also the <i>RowsPerMessage</i> configuration parameter.	decided by the server
<i>NoAssertMessages</i>	This parameter is relevant to the Windows platform only. If set to Yes, the Windows run-time error dialog is not shown.	No

Appendix C. solidDB Command Line Options

Table C.1. solidDB Command Line Options

Option	Description	Examples
-cdir	Changes working directory.	<code>solid -c /data/solid</code>
-d network_name	Disables network name — i.e. instructs the server not to listen for connections on this network name.	<code>solid tcp -d hobbes 1313, shmem -d solid</code>
-f	Starts the server in foreground.	
-h	Displays help.	
-m	Monitors users' messages and SQL statements.	
-nname	Sets the server name.	
-s { start install remove }, name, fullexepath, [autostart]	<p>The Microsoft Windows version of solidDB is by default an icon exe version. However, you can start it as a Windows service by using the option -sstart. When the server is started as a service, it can be started and stopped from the service manager.</p> <p>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. solidDB can also install and remove services using this command line option.</p>	<code>SOLID.EXE -s"install,SOLID, D:\SOLID\SOLID.EXE -sstart -cd:\SOLID"</code> <code>SOLID.EXE -s"install,SOLID, D:\SOLID\SOLID.EXE -sstart -cd:\SOLID,autostart"</code> <code>SOLID.EXE -s"remove,SOLID"</code>

Option	Description	Examples
-Username	See option -x execute or -x exit . If used without the -x option, specifies the username for the database being created.	
-Ppassword	See option -x execute or -x exit . If used without the -x option, specifies the given password for the database being created.	
-Ccatalog	Specify the database catalog	
-E	Encrypt the database	
-Spassword	The database file encryption password.	
-x assert:s	Disables emergency exit dialog.	
-x autoconvert -Ccatalogname	Converts database format to the current format used by solidDB and starts the server process. -Ccatalogname is required to specify the default system catalog name for the database.	
-x convert -Ccatalogname	Converts database format to the current format used by solidDB and starts the server process. -Ccatalogname is required to specify the default system catalog name for the database. The server exits after performing the task.	
-x backupserver	See <i>solidDB High Availability User Guide</i> for information.	
-x disableallmessageboxes	Hides all message windows	
-x decrypt -Spassword	Decrypts the database.	<pre>solid -x decrypt -S dba solid -x decrypt -x keypwdfile:pwd.txt</pre>
-x execute:input file	Prompts for the database administrator's user name and password, creates a new database, executes SQL statements from a file, and exits. The options -U and -P can be used to give the database the administrator's user name and password.	<pre>solid.exe -x execute:init.sql solid.exe -x</pre>

Option	Description	Examples
	The input file must be encoded with a 7-bit or 8-bit character set, such as ASCII or Latin-1.	<code>execute:init.sql</code> <code>-Udba -Pdba</code>
-x executeandnoexit:input file	<p>Prompts for the database administrator's user name and password, creates a new database, executes SQL statements from a file, but does not exit.</p> <p>You can use this command with an existing database provided that you use options -U and -P to give the database the administrator's user name and password.</p> <p>The input file must be encoded with a 7-bit or 8-bit character set, such as ASCII or Latin-1.</p>	<code>solid.exe -x executeandnoexit: init.sql</code> <code>solid.exe -x executeandnoexit: init.sql</code> <code>-Udba -Pdba</code>
-x exit	Prompts for the database administrator's user name and password, creates a new database, and exits. Options -U and -P can be used to give the database administrator's user name and password.	<code>solid.exe -x exit</code> <code>solid.exe -x exit</code> <code>-Udba -Pdba</code>
-x error-msgnostop	Does not wait for user actions on error dialogs.	
-x forcerecovery	Does a forced roll-forward recovery.	
-x hide	Hides the server icon.	
-x ignorecrashed	Ignores log files and reverts to checkpoint.	
-x ignoreerrors	Ignores index errors.	
-x infodbfreefactor	Informs about unused pages. See also: -x reorganize . The server exits after performing the task	
-x inifile: <file-name>	Substitutes an INI file.	
-x listen:<connect-string>	Sets a listening address.	
-x migratehsbg2	This command-line switch has two effects. It instructs the server to accept and convert the existing database (the same effect as the -x autoconvert parameter). Also, it enables the new Secondary to communicate with the old Primary by way of the old replication protocol.	

Option	Description	Examples
	This parameter is needed only when upgrading a server that uses HotStandby.	
-x nologrecovery	With this command-line switch, you can ignore log files during recovery.	
-x outputsql	This command-line switch also prints out the executed SQL commands instead of only printing out the results of each operation.	
-x pathprefix: <dir >	Uses files in the directory <dir>.	
-x pwdfile:file name	The password is read from the file name instead of command-line argument. This way the password can't be seen by running the UNIX command ps .	
-x keypwdfile:file name	The database encryption password is read from the file name instead of command-line argument. This way the password can't be seen by running the UNIX command ps .	
-x recreate_no-confirm	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 returnerroronexit	This command-line switch is used to display return codes for SQL errors and user raised procedure errors. The possible return codes are: Code 60 is returned if the execution of an SQL statement fails. Code 61 is returned if a procedure call returns an error. If several sql statements and/or procedure calls fail during the execution of an SQL script, the returned code is that of the first failure.	
-x stoponerror	This command-line switch is used to force stop and exit solsql immediately when an error is detected.	
-x testblocks	Tests database blocks and exits.	
-x testindex	Tests database index and exits.	
-x testintegrity	Performs a full database integrity test and exits.	
-x version	Displays the server version and exits.	
-?	Help = Usage.	

Option	Description	Examples
-h	Help = Usage.	

Appendix D. Error Codes

This appendix lists error codes that can be generated by the server. Note that some errors specific to options are documented in the option guides.

D.1 Error Categories

Table D.1. solidDB Error Categories

Error category	Description
Synchronization Errors	<p>These errors may be encountered when creating or maintaining the solidDB environment. They occur when using specific solidDB statements, which are Solid SQL extensions.</p> <p>For more information, see Section D.2, “Solid Synchronization Errors”</p>
SQL Errors	<p>These errors are caused by erroneous SQL statements and are detected by the Solid SQL Parser. Administrative actions are not needed.</p> <p>For more information, see Section D.3, “Solid SQL Errors”</p>
Solid SQL API Errors	<p>These errors are caused by erroneous use of the Solid SQL API (SSA). Solid ODBC and JDBC drivers are implemented on this API.</p> <p>For more information, see Section D.4, “Solid SQL API Errors”</p>
Database Errors	<p>These errors are detected by the solidDB and may demand administrative actions.</p> <p>For more information, see Section D.5, “Solid Database Errors”</p>
Executable Errors	<p>These errors are caused by the failure of a solidDB executable or a command line argument related error. They enable implementing intelligent error handling logic in system startup scripts.</p> <p>For more information, see Section D.6, “Solid Executable Errors”</p>
System Errors	<p>These errors are detected by the operating system and demand administrative actions.</p> <p>For more information, see Section D.7, “Solid System Errors”</p>
Table Errors	<p>These errors are caused by erroneous SQL statements and detected by solidDB. Administrative actions are not needed.</p>

Error category	Description
	For more information, see Section D.8, “Solid Table Errors”
Server Errors	<p>These errors are caused by erroneous administrative actions or client requests. They may demand administrative actions.</p> <p>For more information, see Section D.9, “Solid Server Errors”</p>
Communication Errors	<p>These errors are encountered by network problems, faulty configuration of the solidDB software, of ping facility errors. These errors usually demand administrative actions.</p> <p>For more information, see Section D.10, “Solid Communication Errors”</p>
Procedure Errors	<p>These errors are encountered when defining or executing a stored procedure. Administrative actions are not needed.</p> <p>For more information, see Section D.12, “Solid Procedure Errors”</p>
Sorter Errors	<p>These errors are encountered when the external sorter algorithm is solving queries that require ordering rows</p> <p>For more information, see Section D.13, “Solid Sorter Errors”</p>
Solid SpeedLoader Utility (sol-load) Errors	<p>These errors are encountered when running the SpeedLoader utility (solload) to load data from external ASCII files into a solidDB database.</p> <p>For more information, see Section D.14, “Solid SpeedLoader Utility (sol-load) Errors”</p>
Internal Errors	If you receive an internal error, please contact Solid Technical Support.

D.2 Solid Synchronization Errors

Table D.2. Solid Synchronization Errors

Error code	Description
25001	<p>Master cannot save propagated statements.</p> <p>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:</p> <ul style="list-style-type: none"> Master database has exceeded the database size limit. You can increase the database size by changing the <i>FileSpec</i> parameter setting in the solid.ini file. For details on

Error code	Description
	<p>this parameter, read the section called “FileSpec_[1...N] Parameter”. Be sure to restart the server for the new setting to take effect.</p> <ul style="list-style-type: none"> An internal error exists in the database server. If error 25001 occurs even after you have increased the database size, contact Solid Technical support.
25002	Can not save data dictionary statements.
25003	<p>Cannot save SAVE statements.</p> <p>It is not possible to save a "SAVE" statement for later propagation. For example, the following SQL statement returns an error:</p> <pre>SAVE CALL MYPROC(1, 'foo')</pre> <p>solidDB statements that return this error:</p> <pre>SAVE sql_statement</pre>
25004	<p>Dynamic parameters are not supported.</p> <p>Input parameters of a subscription must be given as literals. They cannot be dynamically bound to the statement.</p> <p>solidDB statements that return this error:</p> <pre>DROP SUBSCRIPTION MESSAGE message_name APPEND REFRESH publication_name</pre>
25005	<p>Message <i>message_name</i> is already active.</p> <p>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:</p>

Error code	Description
	<p>MESSAGE <i>message_name</i> BEGIN</p> <p>The message is automatically deleted when the reply of the message has been successfully executed in the replica database.</p> <p>solidDB statements that return this error:</p> <p>MESSAGE <i>message_name</i> APPEND MESSAGE <i>message_name</i> BEGIN MESSAGE <i>message_name</i> DELETE MESSAGE <i>message_name</i> EXECUTE MESSAGE <i>message_name</i> FORWARD MESSAGE GET REPLY</p>
25006	<p>Message <i>message_name</i> not active</p> <p>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.</p> <p>You must first remove the message with MESSAGE <i>message_name</i> DELETE command. Then switch autocommit off and run the script again.</p> <p>solidDB statements that return this error:</p> <p>MESSAGE <i>message_name</i> APPEND <i>synchronization_task</i></p>
25007	<p>Master <i>master_name</i> not found</p> <p>A replica attempts to perform an operation to a master database that cannot be found.</p> <p>solidDB statements that return this error:</p> <p>SET SYNC CONNECT <i>connect_string</i> TO MASTER <i>master_name</i></p>

Error code	Description
	<pre> DROP MASTER <i>master_name</i> IMPORT '<i>filename</i>' SAVE <i>sql_statement</i> </pre>
25009	<p>Replica <i>replica_name</i> not found</p> <p>The replica name specified in a command cannot be found.</p> <p>solidDB statements that return this error:</p> <pre> DROP REPLICA <i>replica_name</i> DROP SUBSCRIPTION <i>publication_name</i>(<i>parameter_list</i>) [FROM REPLICA <i>replica_name</i>] GRANT REFRESH ON <i>publication_name</i> MESSAGE DELETE CURRENT TRANSACTION MESSAGE <i>message_name</i> [FROM REPLICA <i>replica_name</i>] DELETE </pre>
25010	<p>Publication <i>publication_name</i> not found.</p> <p>The publication name of a subscription is incorrect.</p> <p>solidDB statements that return this error:</p> <pre> MESSAGE APPEND REFRESH <i>publication_name</i>(<i>parameter_list</i>) DROP PUBLICATION <i>publication_name</i> EXPORT SUBSCRIPTION <i>publication_name</i> ... REVOKE REFRESH ON <i>publication_name</i>... </pre>
25011	<p>Wrong number of parameters to publication <i>publication_name</i>.</p> <p>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.</p> <p>solidDB statements that return this error:</p>

Error code	Description
	<pre> DROP SUBSCRIPTION <i>publication_name</i> (<i>parameter_list</i>) [FROM REPLICA <i>replica_name</i>] MESSAGE <i>message_name</i> APPEND REFRESH <i>publication_name</i> (<i>parameter_list</i>) </pre>
25012	<p>Message reply timed out.</p> <p>A reply message has not arrived to the replica database within the given timeout period. The reason is that the reply message is not yet ready in the master database. The message needs to be retrieved later using "MESSAGE <i>message_name</i> GET REPLY" command.</p> <p>solidDB statements that return this error:</p> <pre> MESSAGE <i>message_name</i> FORWARD TIMEOUT <i>timeout_in_seconds</i> MESSAGE <i>message_name</i> GET REPLY TIMEOUT <i>timeout_in_seconds</i> </pre>
25013	<p>Message name <i>message_name</i> not found.</p> <p>The message with the given name does not exist. The message name is given when the message is created with command MESSAGE <i>message_name</i> BEGIN. The message name is released when the reply message has been successfully executed in the replica database.</p> <p>Message names must be unique within the replica database.</p> <p>A message can be deleted from the database with command:</p> <pre> MESSAGE <i>message_name</i> [FROM REPLICA <i>replica_name</i>] DELETE </pre> <p>solidDB statements that return this error:</p> <pre> MESSAGE <i>message_name</i> APPEND MESSAGE <i>message_name</i> DELETE MESSAGE <i>message_name</i> END MESSAGE <i>message_name</i> EXECUTE </pre>

Error code	Description
	<pre> MESSAGE <i>message_name</i> FORWARD MESSAGE <i>message_name</i> FROM REPLICA EXECUTE MESSAGE <i>message_name</i> FROM REPLICA <i>replica_name</i> DELETE CURRENT TRANSACTION MESSAGE <i>message_name</i> GET REPLY </pre>
25014	More than one master name found.
25015	<p>Syntax error: <i>error_message</i>, line <i>line_number</i></p> <p>Syntax is not correct.</p> <p>solidDB statements that return this error:</p> <pre> MESSAGE <i>message_name</i> APPEND CREATE PUBLICATION <i>publication_name</i> </pre> <p>Note: See the CREATE PUBLICATION syntax reference for correct syntax.</p>
25016	<p>Message not found, replica id <i>replica_id</i>, message id <i>message_id</i></p> <p>Message not found in master during processing. This can happen if the message is explicitly deleted in master.</p> <p>solidDB statements that return this error:</p> <pre> MESSAGE <i>message_name</i> FORWARD MESSAGE <i>message_name</i> GET REPLY MESSAGE <i>message_name</i> RESTART </pre>
25017	<p>No unique key found for table <i>table_name</i>.</p> <p>The primary key for the table has not been defined.</p> <p>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.</p>

Error code	Description
	<p>solidDB statements that return this error:</p> <pre>ALTER TABLE <i>table_name</i> SET SYNCHISTORY</pre>
25018	<p>Illegal message state.</p> <p>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:</p> <pre>MESSAGE <i>message_name</i> [FROM REPLICA <i>replica_name</i>] DELETE</pre> <p>solidDB statements that return this error:</p> <pre>MESSAGE <i>message_name</i> ...</pre>
25019	<p>Database is not a replica.</p> <p>A synchronization message can only be created in a database that has been registered to be a replica database. See the example code in <i>solidDB SmartFlow Data Replication Guide</i>, which provides information on registering a replica database.</p> <p>solidDB statements that return this error:</p> <pre>DROP MASTER <i>master_name</i> DROP PUBLICATION <i>publication_name</i> REGISTRATION DROP SUBSCRIPTION <i>publication_name</i> ... IMPORT '<i>filename</i>' MESSAGE <i>message_name</i> BEGIN MESSAGE <i>message_name</i> ENDSET SYNC CONNECT '<i>connect_string</i>' TO MASTER <i>master_name</i></pre>
25020	Database is not a master.

Error code	Description
	<p>A command that can be executed only in a master database has been attempted to execute in a non-master database.</p> <p>A database can be set to be a master database of a system by entering the following command:</p> <pre>SET SYNC MASTER YES</pre> <p>solidDB statements that return this error:</p> <pre>ALTER USER <i>replica_user</i> SET MASTER <i>master_name</i> USER MESSAGE <i>message_name</i> FROM REPLICA <i>replica_name</i> RESTART MESSAGE <i>message_name</i> FROM REPLICA <i>replica_name</i> DELETE DROP REPLICA <i>replica_name</i> DROP SUBSCRIPTION <i>subscription_name</i> FROM REPLICA <i>replica_name</i></pre>
25021	<p>Database is not master or replica database.</p> <p>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).</p> <p>solidDB statements that return this error:</p> <pre>CREATE PUBLICATION <i>publication_name</i> ... DROP PUBLICATION <i>publication_name</i> REGISTRATION SET SYNC MAINTENANCE MODE ...; ALTER TABLE <i>table_name</i> SET SYNCHISTORY</pre>
25022	<p>User generated error.</p> <p>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.</p> <p>User can request solidDB to roll back a transaction by setting the following parameters to the bulletin board of the transaction:</p>

Error code	Description
	<pre>PutParam('SYS_ROLLBACK', 'YES') PutParam('SYS_ERROR_CODE', <i>numeric_value_as_string</i>) PutParam('SYS_ERROR_TEXT', <i>error_text_as_string</i>)</pre> <p>If the SYS_ERROR_CODE parameter is not specified or it contains an invalid value, the error number 25022 is returned.</p> <p>solidDB statements that return this error:</p> <pre>MESSAGE <i>message_name</i> FORWARD TIMEOUT <i>timeout_in_seconds</i> MESSAGE <i>message_name</i> GET REPLY TIMEOUT <i>timeout_in_seconds</i></pre>
25023	<p>Replica registration failed.</p> <p>An error has occurred during replica registration.</p> <p>solidDB statements that return this error:</p> <pre>MESSAGE <i>message_name</i> FORWARD TIMEOUT <i>timeout_in_seconds</i> MESSAGE <i>message_name</i> GET REPLY TIMEOUT <i>timeout_in_seconds</i></pre>
25024	<p>Master not defined.</p> <p>No definition for the master exists or the configuration changed during message processing. solidDB was unable to properly initialize the synchronization environment. You can check the master from the replica's system table SYS_SYNC_MASTERS. All successfully registered replicas are found from the master database system table SYS_SYNC_REPLICAS.</p> <p>Note that this error can be produced if you use double quotes rather than single quotes around the master_connect_string in a MESSAGE FORWARD command. solidDB statements that return this error:</p> <pre>IMPORT '<i>filename</i>' MESSAGE <i>message_name</i> FORWARD TO '<i>master_connect_string</i>'</pre>

Error code	Description
	<p>TIMEOUT <i>timeout_in_seconds</i></p> <p>Note: The use of double quotes rather than single quotes around the master_connect_string in can produce this error message.</p> <pre>MESSAGE message_name GET REPLY ... MESSAGE message_name APPEND REFRESH publication_name MESSAGE message_name EXECUTE</pre>
25025	<p>Node name not defined.</p> <p>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:</p> <pre>SET SYNC NODE node_name</pre> <p>solidDB statements that return this error:</p> <pre>DROP PUBLICATION publication_name REGISTRATION MESSAGE message_name APPEND REGISTER REPLICA MESSAGE message_name BEGIN ...</pre>
25026	<p>A user who has not been defined in the master database, attempts to perform a solidDB SQL command.</p> <p>solidDB statements that return this error:</p> <pre>IMPORT 'filename' SAVE sql_statement MESSAGE message_name ...</pre> <p>To resolve this problem, use the correct user ID if there is one. If there is not already a correct user ID, then you have two options:</p>

Error code	Description
	<p>1) Map a master user to the replica userid you are using. (The master user must already have been downloaded from the master to the replica.) To map a master user to a replica user, execute the command:</p> <pre>ALTER USER <i>replica_user</i> SET MASTER <i>master_name</i> USER <i>user_specification</i></pre> <p>2) Add an appropriate user to the master database, and download it with:</p> <pre>MESSAGE <i>message_name</i> APPEND SYNC_CONFIG</pre>
25027	Too long column or parameter value; configured maximum is %Id
25028	<p>Message <i>message_name</i> can include only one system subscription.</p> <p>System subscriptions (REGISTER REPLICA and SYNC_CONFIG) must be kept in separate messages. These tasks must be the only ones of their messages.</p> <p>solidDB statements that return this error:</p> <pre>MESSAGE <i>message_name</i> APPEND REFRESH <i>publication_name</i></pre>
25030	<p>Replica <i>replica_name</i> is already registered.</p> <p>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 <i>replica_name</i>. 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.</p> <p>solidDB statements that return this error:</p> <pre>MESSAGE <i>message_name</i> FORWARD ... MESSAGE <i>message_name</i> GET REPLY ...</pre>

Error code	Description
25031	<p>Transaction is active, operation failed.</p> <p>A replica attempts to process a message when having an active transaction.</p> <p>solidDB statements that return this error:</p> <pre>IMPORT 'filename' MESSAGE message_name FORWARD ... MESSAGE message_name GET REPLY TIMEOUT ... MESSAGE message_name EXECUTE</pre>
25032	<p>All publication SQL statements must return rows.</p> <p>The publication definition contains SQL operations that don't return rows. Only SELECT statements are allowed in the publication.</p> <p>solidDB statements that return this error:</p> <pre>CREATE PUBLICATION publication_name</pre>
25033	<p>Publication <i>publication_name</i> already exists.</p> <p>A publication has been attempted to create with a name that is already in use.</p> <p>solidDB statements that return this error:</p> <pre>CREATE PUBLICATION publication_name</pre>
25034	<p>Message name <i>message_name</i> already exists.</p> <p>Each message must have a name that is unique within the database.</p> <p>solidDB statements that return this error:</p>

Error code	Description
	<p>MESSAGE <i>message_name</i> BEGIN</p>
25035	<p>Message <i>message_name</i> is in use.</p> <p>A solidDB message is locked during an attempt to execute it or delete it. A locked message cannot be re-executed or deleted. If you get this error while attempting to create a new solidDB message, it is probably due to an existing message with the same name. You can check existing messages from the system table SYS_SYNC_REPLICA_MSGINFO in the replica or from the system table SYS_SYNC_MASTER_MSGINFO in the master database.</p> <p>solidDB statements that return this error:</p> <pre> MESSAGE <i>message_name</i> BEGIN MESSAGE <i>message_name</i> END MESSAGE <i>message_name</i> EXECUTE ... MESSAGE <i>message_name</i> FROM REPLICA <i>replica_name</i> DELETE MESSAGE <i>message_name</i> FORWARD TIMEOUT ... MESSAGE <i>message_name</i> GET REPLY TIMEOUT ... </pre>
25036	<p>Publication <i>publication_name</i> not found or publication version mismatch.</p> <p>A publication has been dropped or redefined at master during message processing. Recover by DROP SUBSCRIPTION at replica.</p> <p>solidDB statements that return this error:</p> <pre> IMPORT '<i>filename</i>' MESSAGE <i>message_name</i> FORWARD TIMEOUT ... MESSAGE <i>message_name</i> GET REPLY TIMEOUT ... MESSAGE <i>message_name</i> EXECUTE ... </pre>
25037	<p>Publication column count mismatch in table <i>table_name</i>.</p>

Error code	Description
	<p>Database definitions at master and replica do not match.</p> <p>solidDB statements that return this error:</p> <pre>MESSAGE <i>message_name</i> FORWARD TIMEOUT <i>timeout_in_seconds</i> MESSAGE <i>message_name</i> GET REPLY TIMEOUT <i>timeout_in_seconds</i> MESSAGE <i>message_name</i> EXECUTE</pre>
25038	<p>Table is referenced in publication <i>publication_name</i>; drop or alter operations are not allowed.</p> <p>A table which is referenced in a publication can not be dropped or altered.</p> <p>solidDB statements that return this error:</p> <pre>DROP TABLE <i>table_name</i> ALTER TABLE <i>table_name</i></pre>
25039	<p>Table is referenced in subscription to publication <i>publication_name</i>; drop or alter operations are not allowed.</p> <p>solidDB statements that return this error:</p> <pre>ALTER TABLE <i>table_name</i></pre>
25040	<p>User id <i>user_id</i> is not found.</p> <p>User information has been changed at the replica during message execution.</p> <p>solidDB statements that return this error:</p> <pre>IMPORT '<i>filename</i>' MESSAGE <i>message_name</i> GET REPLY TIMEOUT <i>timeout_in_seconds</i></pre>

Error code	Description
	<p>MESSAGE <i>message_name</i> EXECUTE ...</p> <p>MESSAGE <i>message_name</i> FORWARD ...</p>
25041	<p>Subscription to publication <i>publication_name</i> not found.</p> <p>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.</p> <p>solidDB statements that return this error:</p> <pre> IMPORT 'filename' MESSAGE <i>message_name</i> EXECUTE ... MESSAGE <i>message_name</i> FORWARD ... MESSAGE <i>message_name</i> GET REPLY ... DROP SUBSCRIPTION <i>subscription_name</i> DROP SUBSCRIPTION <i>subscription_name</i> REPLICA <i>replica_name</i> </pre>
25042	<p>Message is too long (<i>number</i> bytes) to forward. Maximum is set to <i>number</i> bytes.</p> <p>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.</p> <p>solidDB statements that return this error:</p> <pre> MESSAGE <i>message_name</i> FORWARD </pre>
25043	<p>Reply message is too long (<i>number</i> bytes). Maximum is set to <i>number</i> bytes.</p> <p>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.</p> <p>solidDB statements that return this error:</p> <pre> MESSAGE <i>message_name</i> GET REPLY </pre>

Error code	Description
25044	<p>SYNC_CONFIG system publication takes only character arguments.</p> <p>In a subscription attempt, publication SYNC_CONFIG was found to have invalid data types for the arguments.</p> <p>solidDB statements that return this error:</p> <pre>MESSAGE <i>message_name</i> APPEND REFRESH SYNC_CONFIG</pre>
25045	Master/replica node support disabled.
25046	<p>Commit and rollback are not supported in propagated transactions.</p> <p>This error is caused when a transaction attempts to execute a COMMIT or ROLLBACK command in the master database. The error is returned to the solidDB server running the procedure. The message containing the procedure will fail.</p>
25047	Parameter info publication not found.
25048	<p>Publication <i>publication_name</i> request info not found.</p> <p>A publication has been dropped while message is being executed.</p> <p>solidDB statements that return this error:</p> <pre>IMPORT '<i>filename</i>' MESSAGE <i>message_name</i> EXECUTE ... MESSAGE <i>message_name</i> FORWARD ... MESSAGE <i>message_name</i> GET REPLY ...</pre>
25049	<p>Referenced table <i>table_name</i> not found in subscription hierarchy.</p> <p>A publication has referenced a table which does not exist.</p> <p>solidDB statements that return this error:</p> <pre>CREATE PUBLICATION <i>publication_name</i> ...</pre>

Error code	Description
25050	Table has no history.
25051	<p>Unfinished messages found.</p> <p>Replica mode has been attempted to be switched off while there are messages either waiting to be forwarded or being executed at master.</p> <p>solidDB statements that return this error:</p> <pre>SET SYNC REPLICA NO</pre>
25052	<p>Failed to set node name to <i>node_name</i>.</p> <p>The <i>node_name</i> may be invalid.</p>
25053	Replica not registered in master.
25054	<p>Table <i>table_name</i> is not set for synchronization history.</p> <p>A table in the master database has the SYNCHISTORY property set, but the corresponding table in the replica does not.</p> <p>solidDB statements that return this error:</p> <pre>IMPORT 'filename' MESSAGE message_name GET REPLY ... MESSAGE message_name FORWARD ...</pre>
25055	<p>Connect information is allowed only when not registered.</p> <p>The connect info in MESSAGE <i>message_name</i> FORWARD TO <i>connect_info</i> options is allowed only if the replica has not yet been registered to the master database.</p> <p>solidDB statements that return this error:</p> <pre>MESSAGE message_name FORWARD TO connect_info options</pre>

Error code	Description
25056	<p>Autocommit not allowed.</p> <p>The solidDB statement must be executed with autocommit mode turned off.</p> <p>solidDB statements that return this error:</p> <pre> All MESSAGE <i>message_name</i> ... statements DROP SUBSCRIPTION <i>subscription_name</i> DROP SUBSCRIPTION <i>subscription_name</i> REPLICA <i>replica_name</i> DROP REPLICA <i>replica_name</i> DROP MASTER <i>master_name</i> EXPORT SUBSCRIPTION IMPORT '<i>filename</i>' </pre>
25057	<p>Already registered to master <i>master_name</i>.</p> <p>The replica database has already been registered to a master database.</p> <p>solidDB statements that return this error:</p> <pre> MESSAGE <i>message_name</i> GET REPLY ... (when registering a replica) MESSAGE <i>message_name</i> FORWARD ... (when registering a replica) </pre>
25058	Missing connect information.
25059	<p>After registration nodename cannot be changed.</p> <p>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.</p> <p>solidDB statements that return this error:</p> <pre> SET SYNC NODE NAME <i>unique_node_name</i> </pre>

Error code	Description
25060	<p>Column <i>column_name</i> does not exist on publication <i>publication_name</i> resultset in table <i>table_name</i>.</p> <p>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.</p> <p>solidDB statements that return this error:</p> <pre>IMPORT 'filename' MESSAGE <i>message_name</i> GET REPLY ... MESSAGE <i>message_name</i> FORWARD ...</pre>
25061	<p>Where condition for table <i>table_name</i> must refer to an outer table of the publication.</p> <p>If a publication contains nested SELECTs, the WHERE clause of the inner SELECT must refer to the outer table of the outer SELECT.</p> <p>solidDB statements that return this error:</p> <pre>CREATE PUBLICATION <i>publication_name</i></pre>
25062	<p>User <i>user_id</i> is not mapped to master <i>user_id</i>.</p> <p>Dropping the user mapping failed because user is not mapped to a given master.</p> <p>solidDB statements that return this error:</p> <pre>ALTER USER <i>replica_user</i> SET MASTER <i>master_name</i> USER</pre>
25063	<p>User <i>user_id</i> is already mapped to master <i>user_id</i>.</p> <p>User is already mapped to a given master.</p> <p>solidDB statements that return this error:</p>

Error code	Description
	ALTER USER <i>replica_user</i> SET MASTER <i>master_name</i> USER
25064	<p>Unfinished message <i>message_name</i> found for replica <i>replica_name</i>.</p> <p>Dropping the replica failed because there are unfinished messages.</p> <p>solidDB statements that return this error:</p> <pre>DROP REPLICA <i>replica_name</i></pre>
25065	<p>Unfinished message <i>message_name</i> found for master <i>master_name</i>.</p> <p>Dropping the master failed because there are unfinished messages.</p> <p>solidDB statements that return this error:</p> <pre>DROP MASTER <i>master_name</i></pre>
25066	<p>Synchronization bookmark <i>bookmark_name</i> already exists.</p> <p>Cannot create synchronization bookmark since the name already exists.</p> <p>solidDB statements that return this error:</p> <pre>CREATE SYNC BOOKMARK</pre>
25067	<p>Synchronization bookmark <i>bookmark_name</i> not found.</p> <p>Bookmark name is not an existing bookmark.</p> <p>solidDB statements that return this error:</p>

Error code	Description
	DROP SYNC BOOKMARK
25068	Export file <i>file_name</i> open failure. Failed to open export file for EXPORT SUBSCRIPTION. solidDB statements that return this error: EXPORT SUBSCRIPTION
25069	Import file <i>file_name</i> open failure. Failed to open import file for IMPORT. solidDB statements that return this error: IMPORT ' <i>filename</i> '
25070	Statements can be saved only for one master in transaction. Statements cannot be saved for multiple masters in one transaction. solidDB statements that return this error: SAVE <i>sql_statement</i>
25071	Not registered to publication <i>publication_name</i> . Replica must be registered to a publication before the publication can be refreshed to the replica. solidDB statements that return this error:

Error code	Description
	<p>DROP PUBLICATION <i>publication_name</i> REGISTRATION MESSAGE <i>message_name</i> APPEND REFRESH <i>publication_name</i></p>
25072	<p>Already registered to publication <i>publication_name</i>.</p> <p>Replica is already registered to a publication.</p> <p>solidDB statements that return this error:</p> <p>MESSAGE <i>message_name</i> APPEND REGISTER REPLICA</p>
25073	Export file can have data only from one master.
25074	<p>User definition not allowed for this operation.</p> <p>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.</p> <p>solidDB statements that return this error:</p> <p>SAVE <i>sql_statement</i> DROP SUBSCRIPTION <i>publication_name</i> (in replica) MESSAGE <i>message_name</i> APPEND REFRESH <i>publication_name</i> MESSAGE <i>message_name</i> APPEND PROPAGATE TRANSACTIONS MESSAGE <i>message_name</i> APPEND REGISTER PUBLICATION MESSAGE <i>message_name</i> APPEND UNREGISTER PUBLICATION MESSAGE <i>message_name</i> EXECUTE (in replica)</p>
25075	Transaction not found.
25076	Only REGISTER REPLICA is allowed in message.
25077	Node name is not valid.
25078	Node name already exists.

Error code	Description
25079	Catalog is master and there are registered replicas. Catalog is not dropped.
25080	Catalog is replica and it is registered to a master. Catalog is not dropped.
25081	Subqueries are not allowed in publication definition.
25082	<p>Node name can not be removed if node is master or replica.</p> <p>Node name cannot be set to NONE on a synchronized master and/or replica catalog.</p> <p>solidDB statements that return this error:</p> <pre>SET SYNC NODE NONE</pre>
25083	Commit block can not be used with Hot StandBy.
25084	Can not save ADMIN COMMAND.
25085	<p>Failed to store blob from message.</p> <p>During synchronization, reading or storing a BLOB (LONG VARCHAR or LONG VARBINARY data) has failed because of an internal error.</p>
25086	Cannot save START statement.
25087	<p>Missing connect information for node '<node_name>'.</p> <p>There is no connect string in the table sys_sync_replicas for the specified replica. Registering a replica doesn't automatically add the connect string into that table if you haven't defined it in the replica's <code>solid.ini</code>. You should define it as shown below:</p> <pre>[Synchronizer] ConnectStrForMaster=tcp replicahost 1316</pre>
25088	Catalog already in maintenance mode. You have set the mode on already.
25089	Not allowed to set maintenance mode off. Someone else has set the mode on, so you cannot set it off.
25090	Catalog already in maintenance mode. Someone else has set the mode on, so you cannot set it off.
25091	Catalog is not in maintenance mode. You tried to set the mode off when it was not on.

Error code	Description
25092	<p>User version strings are not equal in master and replica, operation failed.</p> <p>When the replica executes either of the following commands:</p> <pre>MESSAGE FORWARD MESSAGE GET REPLY</pre> <p>the server checks whether the master and replica sync schema version numbers are equal. If the version numbers are not equal, then the server gives this error. (Note: If neither the master nor the replica has set the version number, then you won't get the error message.)</p>
25093	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	Received illegal message part type.
25095	Message execution aborted.

D.3 Solid SQL Errors

Table D.3. Solid SQL Errors

Error code	Description
SQL Error 1	<p>Parsing error 'syntax error'</p> <p>The SQL parser could not parse the SQL string. Check the syntax of the SQL statement and try again.</p>
SQL Error 2	<p>Table <i>table</i> can not be opened</p> <p>You may not have privileges to access the table and its data.</p>
SQL Error 3	<p>Table <i>table</i> can not be created</p> <p>Table can not be created. You may not have privileges for this operation.</p>
SQL Error 4	<p>Illegal type definition <i>column</i></p> <p>A column type in your CREATE TABLE statement is illegal. Use a legal type for the column.</p>

Error code	Description
SQL Error 5	Table <i>table</i> 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 <i>column</i> 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 <i>view</i> 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.
SQL Error 11	View <i>view</i> cannot be dropped. You cannot drop this view. Only the owner (i.e. the creator) of the view can drop it.
SQL Error 12	Illegal view definition <i>view</i> The view definition is illegal. Check the syntax of the definition.
SQL Error 13	Illegal column name <i>column</i> Column name is illegal. Check that the name is not a reserved name.
SQL Error 14	Call to function <i>function</i> 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

Error code	Description
	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.

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 <i>constant</i> 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

Error code	Description
	Rolling back a transaction failed.
SQL Error 41	Savepoint could not be created A savepoint could not be created.
SQL Error 42	Could not create index <i>index</i> 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 <i>index</i> 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 <i>schema</i> A schema could not be created.
SQL Error 45	Could not drop schema <i>schema</i> 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 <i>expression</i> 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 <i>column</i>

Error code	Description
	You have referenced a column which exists in more than one table. Use syntax <i>table.column</i> to indicate which table you want to use.
SQL Error 51	Non-existent function <i>function</i> You tried to use a function which does not exist.
SQL Error 52	Non-existent cursor <i>cursor</i> You tried to use a cursor which is not created.
SQL Error 53	Function call sequence error A function was called in wrong order. Check the sequence and success of the function calls.
SQL Error 54	Illegal use of a parameter A parameter was used illegally. For example: SELECT * FROM TEST WHERE ? < ?;
SQL Error 55	Illegal parameter value A parameter has an illegal value. Check the type and value of the parameter.
SQL Error 56	Only ANDs and simple condition predicates allowed in UPDATE CHECK All search condition predicates are not supported.
SQL Error 57	Opening the cursor did not succeed Server failed to open a cursor. You may not have cursor open at this moment.
SQL Error 58	Column <i>column</i> 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.

Error code	Description
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 <i>cursor</i> 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 <i>user</i> A user could not be created. You may not have privileges for this operation.
SQL Error 67	Could not alter user <i>user</i> A user could not be altered. You may not have privileges for this operation.
SQL Error 68	Could not drop user <i>user</i> A user could not be dropped. You may not have privileges for this operation.
SQL Error 69	Could not create role <i>role</i> A role could not be created. You may not have privileges for this operation.
SQL Error 70	Could not drop role <i>role</i> 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	Revoking role failed. You may not have privileges for this operation.
Revoke role failed	
SQL Error 73	Comparison of vectors of different length

Error code	Description
	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	<p>Expression * not compatible with aggregate expression</p> <p>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.</p>
SQL Error 75	<p>Illegal reference to table <i>table</i></p> <p>You have tried to reference a table which is not in the FROM list. For example: SELECT T1.* FROM T2.</p>
SQL Error 76	<p>Ambiguous table name <i>table</i></p> <p>You have used the syntax <i>table.column_name</i> ambiguously. For example: SELECT T1.* FROM T1 A,T1 B WHERE A.F1=0;</p>
SQL Error 77	<p>Illegal use of aggregate expression</p> <p>You tried to use aggregate expression illegally. For example: SELECT ID FROM TEST WHERE SUM(ID) = 3;</p>
SQL Error 78	<p>Row fetch failed</p> <p>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.</p>
SQL Error 79	<p>Subqueries not allowed in CHECK constraint</p> <p>You tried to use subquery in a check constraint.</p>
SQL Error 80	<p>Sorting failed</p> <p>External sorter is out of disk space or cache memory. Modify parameters in configuration file <i>solid.ini</i>.</p>
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 <i>statement</i>
SQL Error 85	Incorrect number of values for INSERT
SQL Error 86	Illegal ROWNUM constraint

Error code	Description
SQL Error 88	Subquery not allowed in UPDATE expression Subqueries cannot be used with UPDATE statements.
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.

D.4 Solid SQL API Errors

Table D.4. Solid SQL API Errors

Error code	Description
SSA Error 25200	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.
SSA Error 25201	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.
SSA Error 25202	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.
SSA Error 25203	Invalid transaction operation code This error is given, if an attempt to use an incorrect transaction completion code with the <code>SQLEndTran</code> function (<code>SQL_COMMIT</code> and <code>SQL_ROLLBACK</code> are allowed) is made.
SSA Error 25204	Invalid string or buffer length

Error code	Description
	This error is given, if 0 or any negative buffer size is passed to an ODBC function that requires an application buffer.
SSA Error 25205	Invalid attribute/option identifier This error is given, if an invalid operation code is passed to the <code>SQLSetPos</code> , <code>SQLDriverConnect</code> , <code>SQLFreeStmt</code> and so on.
SSA Error 25206	Connection timeout expired
SSA Error 25207	Invalid cursor state This error is given, for example, if an attempt is made to fetch with a closed cursor.
SSA Error 25208	String data, right truncated This error is given if a string buffer was not big enough.
SSA Error 25209	Datetime field overflow This error is given when updating a date or time column with incorrect data.
SSA Error 25210	COUNT field incorrect This error is given, for example, when trying to pass an extra parameter to an insert statement.
SSA Error 25211	Invalid descriptor index This error is given, for example, when using 0 or negative value as <code>SQLBindParameter</code> column index.
SSA Error 25212	Client unable to establish a connection The ODBC client cannot connect to the server.
SSA Error 25213	Connection name in use This error is given, for example, when trying to reconnect an already connected connection.
SSA Error 25214	Connection does not exist This error is given, for example, when trying to use a closed or not connected connection.
SSA Error 25215	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).

Error code	Description
SSA Error 25216	<p>Connection switch, some session context may be lost</p> <p>This is a TF-1 specific error. A TF-1 connection has encountered a connection switch. The application must roll back the transaction to restore the connection.</p>
SSA Error 25217	<p>Client unable to establish a primary connection</p> <p>This is a TF-1 specific error. The ODBC driver has not been able to establish connection to the primary server, for example, after an application rolled back a transaction after a failover, or if there is no primary server address in the TF-1 connection string (all the reachable servers are secondary).</p>

D.5 Solid Database Errors

Table D.5. Solid Database Errors

Error code	Description
Database Error 10001	<p>Key value is not found.</p> <p>Internal error: a key value cannot be found from the database index.</p>
Database Error 10002	<p>Operation failed.</p> <p>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.</p> <p>You may also receive this error if you try to SET TRANSACTION READ ONLY when the transaction already contains some write operations.</p>
Database Error 10004	<p>Redefinition.</p> <p>Unexpected failure occurred in the database engine.</p> <p>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.</p>
Database Error 10005	<p>Unique constraint violation.</p> <p>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.</p>

Error code	Description
	This error message applies not only to user tables, but also to the system tables. For example, if you try to create a table that has the same name as an existing table, you may see this message. The same applies to other database object names, such as names of users, roles, triggers, etc.
Database Error 10006	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.
Database Error 10007	Transaction is not serializable. The transaction committed is not serializable.
Database Error 10008	Snapshot does not exist.
Database Error 10009	Snapshot is newest.
Database Error 10010	No checkpoint in database. This error occurs when the server has crashed in the middle of creating a new database. Delete the database and log files and try to create the database again.
Database Error 10011	Database headers are corrupted. The headers in the database are corrupted. This may be caused by a disk error or other system failure. Restore the database from the backup.
Database Error 10012	Node split failed. This error is given if the node split of the in-memory database (B+ tree) fails.
Database Error 10013	Transaction is read-only. You tried to do one of the following: 1) Execute conflicting SET TRANSACTION statements, e.g. you executed SET TRANSACTION READ WRITE after you already SET TRANSACTION READ ONLY within the same transaction. 2) Write on a HotStandby database server that is in a Secondary state. 3) Write inside a transaction that is set read-only. Remove the write operation or unset the read-only mode in the transaction.

Error code	Description
	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.
Database Error 10014	<p>Resource is locked.</p> <p>This error occurs when you are trying to use a key value in an index which has been concurrently dropped.</p>
Database Error 10016	<p>Log file is corrupted.</p> <p>One of the log files of the database is corrupted. You can not use these log files. Delete them and start the server again.</p>
Database Error 10017	<p>Too long key value.</p> <p>The maximum length of the key value has been exceeded. The maximum value is one third of the size of the index leaf.</p> <p>If there are blobs (long varchar 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.</p> <p>You can try to:</p> <ol style="list-style-type: none"> 1. Increase the <i>[IndexFile]</i> block size to increase the key value limit 2. Redesign your database to reduce space requirements. Design alternatives include: <ul style="list-style-type: none"> • 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),

Error code	Description
	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.
Database Error 10019	Backup is active You have tried to start a backup when a backup process is already in progress.
Database Error 10020	Checkpoint creation is active. You have tried to start a checkpoint when a checkpoint creation is already in progress.
Database Error 10021	Failed to delete log file. The deletion of a log file in making a backup has failed. Reasons for the failure can be: <ul style="list-style-type: none"> • The log file has already been deleted from the operating system. • The log file has a read-only attribute.
Database Error 10023	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. The log file in the database directory is from another solidDB database. Copy the correct log files to the database directory.
Database Error 10024	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.
Database Error 10026	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 <i>AbortTimeOut</i> in <i>SRV</i> section. The default value is 120 minutes.
Database Error 10027	No active search.

Error code	Description
	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.
Database Error 10028	Referential integrity violation, foreign key values exist. You tried to delete a row that is referenced from a foreign key.
Database Error 10029	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.
Database Error 10030	Backup directory ' <i>directory name</i> ' does not exist. Backup directory is not found. Check the name of the backup directory.
Database Error 10031	Transaction detected a deadlock, transaction is rolled back. Deadlock detected. If necessary, begin transaction again.
Database Error 10032	Wrong database block size specified. The block size of the database file differs from the blocksize given in the configuration file <code>solid.ini</code> .
Database Error 10033	Primary key unique constraint violation. Your primary key definition is not unique.
Database Error 10034	Sequence name <i>sequence</i> conflicts with an existing entity. Choose a unique name for a sequence. The specified name is already used.
Database Error 10035	Sequence does not exist. Check the name of the sequence.
Database Error 10036	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.
Database Error 10037	Can not store sequence value, the target data type is illegal. The valid target data types are BIGINT, INTEGER, and BINARY.
Database Error 10038	Illegal column value for descending index. Corrupted data found in descending index. Drop the index and create it again.

Error code	Description
Database Error 10040	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.
Database Error 10041	Database is read-only.
Database Error 10042	Database index check failed, the database file is corrupted.
Database Error 10043	Database free block list corrupted, same block twice in free list.
Database Error 10044	Primary key can not contain blob attributes.
Database Error 10045	This database is a HotStandby secondary server, the database is read only.
Database Error 10046	Operation failed, data dictionary operation is active. Wait and try again.
Database Error 10047	Replicated transaction is aborted.
Database Error 10048	Replicated transaction contains schema changes, operation failed.
Database Error 10049	Slave server not available any more, transaction aborted
Database Error 10050	Replicated row contains BLOb columns that cannot be replicated.
Database Error 10051	Log file is corrupted.
Database Error 10052	Cannot convert an abnormally closed database. Please use the old solidDB database version to recover the database first.
Database Error 10053	Table is read only.
Database Error 10054	Opening the database file failed. Probably another solidDB process is already running in the same directory.
Database Error 10055	Too little cache memory has been specified for the solidDB process.

Error code	Description
Database Error 10056	Cannot open <i>database file</i> . <i>Error text (number)</i> . Most likely the solidDB process does not have correct access rights to the database file.
Database Error 10057	The database is irrevocably corrupted. Revert to the latest backup.
Database Error 10058	The internal database file format version (<i>number</i>) does not match with the solidDB version. Possible causes for this error include: <ul style="list-style-type: none"> • a version of solidDB that is too old is used with this database • the database has been corrupted
Database Error 10059	The internal header version (<i>number</i>) does not match with the solidDB version. Possible causes for this error include: <ul style="list-style-type: none"> • a version of solidDB that is too old is used with this database • the database has been corrupted
Database error 10060	Cannot perform roll-forward recovery in read-only mode. Read-only mode can be specified in 3 ways. To restart Solid in normal mode, verify that: <ul style="list-style-type: none"> • solidDB process is not started with command-line option -x read only • <code>solid.ini</code> does not contain the following parameter setting: <pre>[General] ReadOnly=yes</pre> • license file does not have read-only limitation
Database error 10061	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

Error code	Description
	<p>users. To allocate more cache memory, set the following <code>solid.ini</code> parameter to a higher value:</p> <pre>[IndexFile] CacheSize=cache_size_in_bytes</pre> <p>NOTE: Allocated cache memory size should not exceed the amount of physical memory.</p>
Database error 10062	<p>Failed to write to <i>log filename</i> at <i>offset</i>.</p> <p>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.</p>
Database error 10063	<p>Cannot create new log filename because such a file already exists in the log file directory.</p> <p>Probably your log file directory also contains logs from some other database. <code>solidDB</code> process cannot continue until invalid log files are removed from the log file directory. Remove <i>log filename</i> and all other log files with greater sequence numbers.</p>
Database error 10064	<p>Illegal log file name template.</p> <p>Most likely, the log file name template specified in:</p> <pre>[Logging] FileNameTemplate=name</pre> <p>contains too few or too many sequence number digit positions. There should be at least 4 and at most 10 digit positions.</p>
Database error 10065	<p>Unknown log write mode. Please, re-check the configuration parameter.</p>
Database error 10066	<p>Cannot open <i>log filename</i>. Check the following log file name template in <code>solid.ini</code>:</p> <pre>[Logging] FileNameTemplate=name</pre> <p>and verify that:</p> <ul style="list-style-type: none"> • it can be expanded into a valid file name in this environment

Error code	Description
	<ul style="list-style-type: none"> solidDB process has appropriate privileges to the log files directory.
Database error 10067	<p>Cannot create database because old <i>log filename</i> exists in the log files directory.</p> <p>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.</p>
Database error 10068	<p>Roll-forward recovery cannot be performed because the configured log file <i>block size number</i> does not match with <i>block size number</i> of existing filename.</p> <p>To enable recovery, edit <code>solid.ini</code> to include parameter setting:</p> <pre>[Logging] BlockSize=blocksize in bytes</pre> <p>and restart the solidDB process. After successful recovery, you can change the log file block size by performing these steps:</p> <ol style="list-style-type: none"> 1. Shut down the solidDB process. 2. Remove old log files. 3. Edit new block size into <code>solid.ini</code>. 4. Restart solidDB.
Database error 10069	Roll-forward recovery failed because <i>relation id number</i> was not found. Database has been irrevocably corrupted. Please restore the database from the last backup.
Database error 10070	Roll-forward failed because <i>relation id number</i> was not found. Database has been irrevocably corrupted. Please restore the database from the latest backup.
Database error 10071	Please restore the database from the latest backup.
Database error 10072	Database operation failed because of the file I/O problem.
Database error 10073	Database is inconsistent. Illegal index block type <i>size</i> , <i>address</i> , <i>routine</i> , <i>reachmode</i> . Please restore the database from the latest backup.
Database error 10074	Roll-forward recovery failed. Please revert to the latest backup.

Error code	Description
Database error 10075	<p>The database you are trying to use has been originally created with different database block size settings than your current settings.</p> <p>Edit the <code>solid.ini</code> file to contain the following parameter setting:</p> <pre>[IndexFile] BlockSize=blocksize in bytes</pre>
Database error 10076	<p>Roll-forward recovery failed because <i>tablename</i> or <i>viewname</i> is redefined in the log filename.</p> <p>Possible causes for this error include:</p> <ul style="list-style-type: none">• another solidDB process is using the same log file directory• old log files are present in the log file directory <p>solidDB process cannot use this corrupted log file to recover. In order to continue, you have the following alternatives:</p> <ol style="list-style-type: none">1. Revert to the last backup2. Revert to the last checkpoint3. Revert to the last committed transaction within the last valid log file
Database error 10077	<p>No base catalog given for database conversion (use -C <i>catalogname</i>)</p> <p>A database's base catalog must be provided when converting the database to a new format.</p>
Database error 10086	<p>Deleted row not found.</p> <p>A key value being deleted cannot be found in the b-tree. This is an internal error.</p>
Database error 10090	<p>Data dictionary operation in a newer transaction.</p> <p>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.</p>
Database error 10091	<p>Backup detected a log file with wrong block size, backup aborted.</p>

Error code	Description
Database error 10092	HotStandby cannot operate when logging is disabled.
Database error 10093	HotStandby migration is not possible if Hotstandby is not configured.
Database error 10094	Only %d cache pages configured for M-table usage, at least %d needed.
Database error 10095	Cursor is closed after isolation change. The current cursor is closed, because its isolation level has been changed.
Database error 10096	Only <kilobytes> kilobytes configured for M-table checkpointing, at least <kilobytes>KB needed. Not enough memory has been configured for the M-table.
Database error 10098	Incrementing sequence <i>sequence_name</i> failed.
Database error 10099	Encryption password has not been given for encrypted database.
Database error 10100	Incorrect password has been given for encrypted database.
Database error 10101	Unknown encryption algorithm.
Database error 10104	Database is not created using solidDB Storage Engine for MySQL Prototype. Cannot open database.
Database error 16501	New row value too large for M-table.
Database error 16502	BLObs are not supported in M-tables.
Database error 16503	Serializable isolation level is not supported in M-tables.
Database error 16504	Memory for M-tables is running low, inserts to M-tables disallowed.
Database error 16505	Ran out of memory for M-tables, updates and inserts to M-tables disallowed.
Database error 16506	Too small configured <i>MME.ImdbMemoryLimit</i> to start server.

Error code	Description
Database error message 30218	Quick merge stopped.

D.6 Solid Executable Errors

Table D.6. Solid Executable Errors

Error code	Description
Executable Error 10	Failed to open database
Executable Error 11	Failed to connect to database
Executable Error 12	Database test failed
Executable Error 13	Database fix failed
Executable Error 14	License error
Executable Error 15	Database must be converted
Executable Error 16	Database does not exist
Executable Error 17	Database exists
Executable Error 18	Database not created
Executable Error 19	Database create failed
Executable Error 20	Communication init failed
Executable Error 21	Communication listen failed
Executable Error 22	Service operation failed
Executable Error 23	Failed to open all the defined database files.
Executable Error 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.

D.7 Solid System Errors

Table D.7. Solid System Errors

Error code	Description
System Error 11000	<p>File open failure.</p> <p>The server is unable to open the database file. Reason for the failure can be:</p> <ul style="list-style-type: none"> • 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. <p>Correct the error and try again.</p>
System Error 11001	<p>File write failure.</p> <p>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.</p>
System Error 11002	<p>File write failed, disk full.</p> <p>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 <i>FileSpec_[1-N]</i> parameter in <i>IndexFile</i> section.</p>
System Error 11003	<p>File write failed, configuration exceeded.</p> <p>Writing to the database file failed, because the maximum database file size set in <i>FileSpec_[1-N]</i> parameter is exceeded.</p>
System Error 11004	<p>File read failure.</p> <p>An error occurred reading a file. This may indicate a disk error in your system.</p>
System Error 11005	<p>File read beyond end of file.</p> <p>This error is given, if the file EOF is reached during the read operation.</p>
System Error 11006	<p>File read failed, illegal file address.</p> <p>An error occurred reading a file. This may indicate a disk error in your system.</p>
System Error 11007	File lock failure.

Error code	Description
	The server failed to lock the database file.
System Error 11008	File unlock failure. The server failed to unlock a file.
System Error 11009	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.
System Error 11010	Too long file name. Filename specified in parameter <i>FileSpec_[1-N]</i> is too long. Change the name to a proper file name.
System Error 11011	Duplicate file name specification. Filename specified in parameter <i>FileSpec_[1-N]</i> is not unique. Change the name to a proper file name.
System Error 11012	License information not found, exiting from solidDB Check the existence of your <code>solid.lic</code> file.
System Error 11013	License information is corrupted. Your <code>solid.lic</code> file has been corrupted.
System Error 11014	Database age limit of evaluation license expired.
System Error 11015	Evaluation license expired.
System Error 11016	License is for different CPU architecture.
System Error 11017	License is for different OS environment.
System Error 11018	License is for different version of this OS.
System Error 11019	License is not valid for this server version.
System Error 11020	License information is corrupted.
System Error 11021	Problem with Your license, please contact Solid Information Technology Ltd. immediately.
System Error 11022	Desktop license is only for local protocol communication, cannot use protocol protocol for listening.
System Error 11023	Internal binary stream error.

Error code	Description
	This error is given if read or write fails when handling a binary stream object.
System Error 11024	Desktop license is only for local communication, cannot use name name for listening.
System Error 11025	License file <i>filename</i> 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.
System Error 11026	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.
System Error 11027	No such parameter section <i>section</i> . Parameter was not found from the specified section in the <code>solid.ini</code> file.
System Error 11028	No such parameter <i>section.name</i> . Parameter does not exist.
System Error 11029	Not allowed to set parameter value. User is not allowed to set the parameter value.
System Error 11030	Cannot set values to multiple parameters. Only one parameter can be set at one time.
System Error 11031	Illegal type for parameter. Parameter type is illegal.
System Error 11032	Cannot set new value for parameter <i>section.name</i> . A new value cannot be set for the parameter.
System Error 11033	Parameter is read-only.
System Error 11034	File remove failure.
System Error 11035	Value for parameter is smaller than minimum value.
System Error 11036	Value for parameter is bigger than maximum value.
System Error 11037	Value for parameter is invalid.
System Error 11038	File specification exceeds the database address space.

Error code	Description
System Error 11039	File specification exceeds the database address space. This error is given if solidDB attempts to use a file, whose given size is larger than the size that solidDB can use.
System Error 11040	Password file cannot be opened. This error is given if solidDB cannot find the database password file.
System Error 11041	No password found in password file. This error is given if the database password is not in the password file.

D.8 Solid Table Errors

Table D.8. Solid Table Errors

Error code	Description
Table Error 13001	Illegal character constant constant. An illegal character constant was found in the SQL statement.
Table Error 13002	Type CHAR not allowed for arithmetic. You have entered a calculation having a character type constant. Character constants are not supported in arithmetic.
Table Error 13003	Aggregate function 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);
Table Error 13004	Illegal aggregate function <i>parameter</i> parameter. An illegal parameter has been given to an aggregate function. Aggregate function parameters can only be column names or numbers.
Table Error 13005	SUM and AVG not supported for CHAR type. Aggregate functions SUM and AVG are not supported for character type parameters.
Table Error 13006	SUM or AVG not supported for DATE type.

Error code	Description
	Aggregate functions SUM and AVG are not supported for date type parameters.
Table Error 13007	Function <i>function</i> is not defined. The function you tried to use is not defined.
Table Error 13008	Illegal parameter to ADD function.
Table Error 13009	Division by zero. A division by zero has occurred.
Table Error 13011	Table <i>table</i> does not exist. You have referenced a table which does not exist or you do not have REFERENCES privilege on the table.
Table Error 13013	Table name <i>table</i> conflicts with an existing entity. Choose a unique name for a table. The specified name is already used.
Table Error 13014	Index <i>index</i> does not exist. You have referenced an index which does not exist.
Table Error 13015	Column <i>column</i> does not exist on table <i>table</i> . You have referenced a column in a table which does not exist.
Table Error 13018	Join table is not supported Joined tables are not supported in this version of solidDB.
Table Error 13019	Transaction savepoints are not supported. Transaction savepoints are not supported in this version of solidDB.
Table Error 13020	Default values are not supported. Default column values are not supported in this version of solidDB.
Table Error 13022	Descending keys are not supported. Descending keys are not supported in this version of solidDB.
Table Error 13023	Schema is not supported. Schema is not supported in this version of solidDB.
Table Error 13025	Update through a cursor with no current row.

Error code	Description
	You have tried to update using a cursor, but you do not have a current row in the cursor.
Table Error 13026	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.
Table Error 13028	View <i>view_name</i> does not exist. You have referenced a view which does not exist.
Table Error 13029	View name <i>view_name</i> conflicts with an existing entity. Choose a unique name for a view. The specified name is already used.
Table Error 13030	No value specified for NOT NULL column <i>column</i> . You have not specified a value for a column which is defined NOT NULL.
Table Error 13031	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.
Table Error 13032	Illegal type <i>type</i> . You have tried to create a table with a column having an illegal type.
Table Error 13033	Illegal parameter <i>parameter</i> for type <i>type</i> . The type of the parameter you entered is illegal in this column.
Table Error 13034	Illegal constant <i>constant</i> . You have entered an illegal constant.
Table Error 13035	Illegal INTEGER constant <i>constant</i> . You have entered an illegal integer type constant. Check the syntax of the statement and try again.
Table Error 13036	Illegal DECIMAL constant <i>constant</i> . You have entered an illegal decimal type constant. Check the decimal number and try again.
Table Error 13037	Illegal DOUBLE PREC constant <i>constant</i> .

Error code	Description
	<p>Typically, this is a general parse error. The SQL statement may contain a syntax error <i>before</i> the constant. As a last resort, the parser has attempted to parse a DOUBLE PREC constant, but has failed.</p> <p>This error also occurs if you entered an illegal double precision type constant.</p> <p>(More specifically, this error occurs when a space is placed between the asterisk and the closing parenthesis ("*") in an optimizer hint.)</p> <p>In any of these cases, be sure to check the syntax of the statement and try again.</p>
Table Error 13038	<p>Illegal REAL constant <i>constant</i>.</p> <p>You have entered an illegal real type constant. Check the real number and try again.</p>
Table Error 13039	<p>Illegal assignment.</p> <p>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.</p>
Table Error 13040	<p>Aggregate <i>function</i> function is not defined.</p> <p>The aggregate function you tried to use is not supported.</p>
Table Error 13041	<p>Type DATE not allowed for arithmetic.</p> <p>DATE type columns or constants are not allowed in arithmetic.</p>
Table Error 13042	<p>Power arithmetic not allowed for NUMERIC and DECIMAL data type.</p> <p>Decimal and numeric data types do not support power arithmetic.</p>
Table Error 13043	<p>Illegal date constant <i>constant</i>.</p> <p>A date constant is illegal. The correct form for date constants is: YYYY-MM-DD.</p>
Table Error 13046	<p>Illegal user name <i>user</i>.</p> <p>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 '_'.</p>
Table Error 13047	No privileges for operation.

Error code	Description
	<p>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.</p> <p>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.)</p>
Table Error 13048	<p>No grant option privilege for entity name.</p> <p>You have no privileges to grant privileges for the entity.</p>
Table Error 13049	<p>Column privileges cannot be granted WITH GRANT OPTION</p> <p>Granting column privileges WITH GRANT OPTION is not supported in this version of solidDB.</p>
Table Error 13050	<p>Too long constraint value.</p> <p>Maximum constraint length has been exceeded. Maximum constraint length is 255 characters.</p>
Table Error 13051	<p>Illegal column name <i>column</i>.</p> <p>You have tried to create a table with an illegal column name.</p>
Table Error 13052	<p>Illegal comparison operator operator for a pseudo column <i>column</i>.</p> <p>You have tried to use an illegal comparison operator for a pseudo column. Legal comparison operators for pseudo columns are: equality '=' and non-equality '<>'.</p>
Table Error 13053	<p>Illegal data type for a pseudo column.</p> <p>You have tried to use an illegal data type for a pseudo column. Data type of pseudo columns is BINARY.</p>
Table Error 13054	<p>Illegal pseudo column data, maybe data is not received using pseudo column.</p>

Error code	Description
	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.
Table Error 13055	Update not allowed on pseudo column. Updates are not allowed on pseudo columns.
Table Error 13056	Insert not allowed on pseudo column. Inserts are not allowed on pseudo columns.
Table Error 13057	Index name <i>index</i> already exists. You have tried to create an index, but an index with the same name already exists. Use another name for the index.
Table Error 13058	Constraint checks were not satisfied on column <i>column</i> . Column has constraint checks which were not satisfied during an insert or update.
Table Error 13059	Reserved system name <i>name</i> . You tried to use a name which is a reserved system name such as PUBLIC and SYS_ADMIN_ROLE.
Table Error 13060	User name <i>user</i> not found. You tried to reference a user name which is not created.
Table Error 13061	Role name <i>role</i> not found. You tried to reference a role name which is not created.
Table Error 13062	Admin option is not supported. Admin option is not supported in this version of solidDB.
Table Error 13063	Name <i>name</i> 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 HOBBS and a role named HOBBS.
Table Error 13064	Not a valid user name <i>user</i> .

Error code	Description
	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 '_'.
Table Error 13065	Not a valid role name <i>role</i> . You tried to create an invalid role 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 '_'.
Table Error 13066	User <i>user</i> not found in role <i>role</i> . You tried to revoke a role from a user and the user did not have that role.
Table Error 13067	Too short password. You have entered a too short password. Password length must be at least 3 characters.
Table Error 13068	Shutdown is in progress. You are unable to complete this operation, because server shutdown is in progress.
Table Error 13070	Numerical overflow. A numerical overflow has occurred. Check the values and types of numerical variables.
Table Error 13071	Numerical underflow. A numerical underflow has occurred. Check the values and types of numerical variables.
Table Error 13072	Numerical value out of range. A numerical value is out of range. Check the values and types of numerical variables.
Table Error 13073	Math error. A mathematical error has occurred. Check the mathematics in the statement and try again.
Table Error 13074	Illegal password. You have tried to enter an illegal password.

Error code	Description
Table Error 13075	<p>Illegal role name <i>role</i>.</p> <p>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 '_'.</p>
Table Error 13077	<p>Last column can not be dropped.</p> <p>You have tried to drop the final column in a table. This is not allowed; at least one column must remain in the table.</p>
Table Error 13078	<p>Column already exist on table.</p> <p>You have tried to create a column which already exists in a table.</p>
Table Error 13079	<p>Illegal search constraint.</p> <p>Check the search engine. There may be mismatch between data types.</p>
Table Error 13080	<p>Incompatible types, can not modify column <i>column</i> from type <i>type</i> to type <i>type</i>.</p> <p>You have tried to modify column to a data type that is incompatible with the original definition, such as VARCHAR and INTEGER</p>
Table Error 13081	<p>Descending keys are not supported for binary columns.</p> <p>You can not define a descending key for a binary column.</p>
Table Error 13082	<p>Function <i>function</i>: parameter * not supported.</p> <p>You can not use parameter star (*) with ODBC Scalar Functions.</p>
Table Error 13083	<p>Function <i>function</i>: Too few parameters.</p> <p>The function expects more parameters. Check the function call.</p>
Table Error 13084	<p>Function <i>function</i>: Too many parameters.</p> <p>The function expects fewer parameters. Check the function call.</p>
Table Error 13085	<p>Function <i>function</i>: Run-time failure.</p> <p>An error was detected during the execution of the function. Check the parameters.</p>
Table Error 13086	<p>Function <i>function</i>: type mismatch in parameter parameter number.</p>

Error code	Description
	An erroneous type of parameter was detected in the given position of the function call. Check the function call.
Table Error 13087	Function <i>function</i> : 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.
Table Error 13088	No primary key for table.
Table Error 13090	Foreign key column <i>column</i> 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.
Table Error 13091	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.
Table Error 13092	Event name <i>event</i> conflicts with an existing entity. Choose a unique name for an event. The specified name is already used.
Table Error 13093	Event <i>event</i> does not exist. You referenced a nonexistent event. Check the name of the event.
Table Error 13094	Duplicate column <i>column</i> in primary key definition. Duplicate columns are not allowed in a table-constraint-definition. Remove duplicate columns from the definition.
Table Error 13095	Duplicate column <i>column</i> in unique constraint definition. Duplicate columns are not allowed in a table-constraint-definition. Remove duplicate columns from the definition.
Table Error 13096	Duplicate column <i>column</i> in index definition. Duplicate columns are not allowed in CREATE INDEX statement. Remove duplicate columns.
Table Error 13097	Primary key columns must be NOT NULL.

Error code	Description
	Error in a <i>column_constraint_definition</i> . Define primary key columns NOT NULL. For example: CREATE TABLE DEPT (DEPTNO INTEGER NOT NULL, DNAME VARCHAR, PRIMARY KEY(DEPTNO));
Table Error 13098	Unique constraint columns must be NOT NULL. Error in a <i>column_constraint_definition</i> . Define unique columns NOT NULL. For example: CREATE TABLE DEPT4 (DEPTNO INTEGER NOT NULL, DNAME VARCHAR, UNIQUE(DEPTNO));
Table Error 13099	No REFERENCES privileges to referenced columns in table <i>table</i> . You do not have privileges to reference to the table.
Table Error 13100	Illegal table mode combination. You have defined an illegal combination of concurrency control settings. For example, this message occurs 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 (Currently, in-memory tables must use pessimistic concurrency control.)
Table Error 13101	Only execute privileges can be used with procedures.
Table Error 13102	Execute privileges can be used only with procedures.
Table Error 13103	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 the himself (i.e. to the DBA).
Table Error 13104	Sequence name <i>sequence</i> conflicts with an existing entity. Choose a unique name for a sequence. The specified name is already used.
Table Error 13105	Sequence <i>sequence</i> does not exist. You referenced a nonexistent sequence. Check the name of sequence.
Table Error 13106	Foreign key reference exists to table <i>table</i> .
Table Error 13107	Illegal set operation. You tried to execute a non-existent set operation.

Error code	Description
Table Error 13108	Comparison between incompatible types <i>datatype</i> and <i>datatype</i> .
Table Error 13109	There are schema objects for this user, drop failed
Table Error 13110	NULL values given for NOT NULL column <i>column</i> .
Table Error 13111	<p>Ambiguous entity name <i>name</i>.</p> <p>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.</p> <p>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:</p> <p>sales_catalog.jan_wong_schema.table.1</p>
Table Error 13112	Foreign keys are not supported with main memory tables.
Table Error 13113	Illegal arithmetic between types <i>datatype</i> and <i>datatype</i> .
Table Error 13114	String operations are not allowed on values stored as BLOBs or CLOBs.
Table Error 13115	<p>Function <i>function_name</i>: Too long value (stored as CLOB) in parameter <i>parameter</i>.</p> <p>The parameter value was stored as CLOB and cannot be used with a function.</p>
Table Error 13116	<p>Column <i>column_name</i> specified more than once.</p> <p>Column was specified more than once in the GRANT or REVOKE statement.</p>
Table Error 13117	<p>Wrong number of parameters</p> <p>Wrong number of parameters when converting subscription parameters to base publication parameter types.</p>
Table Error 13118	<p>Column privileges are supported only for base tables.</p> <p>Column privileges are allowed only for base tables; they cannot be used, for example, for views.</p>
Table Error 13119	<p>Types <i>column_type</i> and <i>column_type</i> are not union compatible.</p> <p>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.</p>

Error code	Description
	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 Table error 13119.
Table Error 13120	Too long entity name ' <i>entity_name</i> '. Entity name is too long, maximum entity name is 254 characters.
Table Error 13121	Too many columns, maximum number of columns per table is <i>value</i> . Note that the maximum number of columns may be less if each column requires a large number of bytes.
Table Error 13122	Operation is not supported for a table with sync history. Operation is not supported because the table has synchronization history defined.
Table Error 13123	Table ' <i>table_name</i> ' is not empty. Some operations are allowed only for empty tables.
Table Error 13124	User id <i>user_id</i> not found. Internal user id was not found; the user may have been dropped.
Table Error 13125	Illegal LIKE pattern ' <i>pattern</i> '. Illegal like pattern was given as a search constraint.
Table Error 13126	Illegal type <i>datatype</i> for LIKE pattern. Only CHAR and WCHAR allowed for LIKE search constraints.
Table Error 13127	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.
Table Error 13128	LIKE predicate failed because value is too long. LIKE predicate failed because the column value is stored as a CLOB.
Table Error 13129	LIKE Predicate failed because pattern is too long.

Error code	Description
	LIKE predicate failed because pattern value is stored as a CLOB.
Table Error 13130	<p>Illegal type <i>datatype</i> for LIKE ESCAPE character.</p> <p>Like ESCAPE character must be CHAR or WCHAR type.</p>
Table Error 13131	<p>Too many nested triggers.</p> <p>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:</p> <p>[SQL] MaxNestedTriggers=n</p>
Table Error 13132	<p>Too many nested procedures.</p> <p>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:</p> <p>[SQL] MaxNestedProcedures=n</p>
Table Error 13133	<p>Not a valid license for this product.</p> <p>The license file is for another Solid product.</p>
Table Error 13134	<p>Operation is allowed only for base tables.</p> <p>Given operation is available only for base tables.</p>
Table Error 13137	<p>Illegal grant/revoke mode</p> <p>Grant or revoke mode is not allowed for given database objects.</p>
Table Error 13138	Index <i>index_name</i> given in index hint does not exist.

Error code	Description
	Index name given in optimizer hint is not found for a table.
Table Error 13139	Catalog <i>catalog_name</i> does not exist. Catalog name is not a valid catalog.
Table Error 13140	Catalog <i>catalog_name</i> already exists. Catalog name is an existing catalog.
Table Error 13141	Schema <i>schema_name</i> does not exist. Schema name is not a valid schema.
Table Error 13142	Schema <i>schema_name</i> already exists. Schema name is an existing schema.
Table Error 13143	Schema <i>schema_name</i> is an existing user. Schema name specifies an existing user name.
Table Error 13144	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.
Table Error 13145	Sync parameter not found. Parameter name given in command SET SYNC PARAMETER name NONE is not found.
Table Error 13146	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.
Table Error 13147	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.
Table Error 13148	There are objects for this schema, drop failed.
Table Error 13149	There are objects for this catalog, drop failed.
Table Error 13150	Index can be created only into same catalog and schema as the base table.

Error code	Description
Table Error 13151	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.
Table Error 13152	There are objects for this user, drop failed.
Table Error 13153	Can not remove last administrator.
Table Error 13154	Name cannot be an empty string.
Table Error 13155	Column <column name> already exists on view <view name> The view definition contains the same column name twice.
Table Error 13156	Column attributes already exists on view.
Table Error 13157	Current schema cannot be dropped.
Table Error 13158	Current user cannot be dropped.
Table Error 13160	Cannot alter table name because it is referenced in trigger(s). Altering the name of the table would prevent the trigger from working properly.
Table Error 13161	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 UPDATE 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.
Table Error 13162	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 UPDATE 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.
Table Error 13163	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.
Table Error 13164	Transaction is active, operation failed.

Error code	Description
Table Error 13165	Can't fetch previous row from an M-table. This message can occur only when fetching rows from in In-Memory table ("M-table") by using solidDB's low-level SA API.
Table Error 13166	License does not allow accessing M-tables You will get this error message if you try to create an in-memory table and you do not have a license that allows you to do this. Generally, you need a license for BoostEngine to create in-memory tables; a license for FlowEngine or EmbeddedEngine is not sufficient.
Table Error 13167	Only M-tables can be transient.
Table Error 13168	Transient tables can not be set temporary.
Table Error 13169	Temporary tables can not be set transient.
Table Error 13170	Only M-tables can be temporary.
Table Error 13171	Foreign key constraints between D- and M-tables are not supported.
Table Error 13172	A persistent table can not reference a transient table. For more details, see the discussion on page B-67 in solidDB SQL Guide.
Table Error 13173	A persistent table can not reference a temporary table. For more details, see the discussion on page B-67 in solidDB SQL Guide.
Table Error 13174	A transient table can not reference a temporary table. For more details, see the discussion on page B-67 in solidDB SQL Guide.
Table Error 13175	A reference between temporary and non-temporary table is not allowed.
Table Error 13176	Cannot change STORE for a table with sync history.
Table Error 13177	Cannot define UNIQUE constraint with duplicated or implied restriction.
Table Error 13178	Constraint not found.
Table Error 13179	Foreign key actions other than restrict are not supported.
Table Error 13180	Constraint name already exists.
Table Error 13181	Constraint check fails on existing data.
Table Error 13182	Added column with NOT NULL must have a non-NULL default.
Table Error 13183	Index is referenced by foreign key, it cannot be dropped.
Table Error 13184	Primary key not found for table. Cannot define foreign key.

Error code	Description
Table Error 13185	Cannot set NOT NULL on column that already has NULL value.
Table Error 13186	Cannot drop NOT NULL on column that is used as part of unique key.
Table Error 13187	The cursor cannot continue accessing M-tables after the transaction has committed or aborted. The statement must be re-executed.
Table Error 13188	Foreign key refers to itself.
Table Error 13189	Positioning is not supported for M-tables.
Table Error 13190	Definition in file is not valid.
Table Error 13191	Parameter setting in file conflicts with the setting in database.
Table Error 13193	<p>Foreign key creates update dependency loop.</p> <p>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.</p> <p>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.</p>
Table Error 13194	Can not drop a table that is part of a foreign key
Table Error 13195	Update failed, READ COMMITTED isolation requires FOR UPDATE
Table Error 13196	Delete failed, READ COMMITTED isolation requires FOR UPDATE
Table Error 13197	M-tables are not supported
Table Error 13198	Commit and rollback are not allowed inside function.
Table Error 13199	<p>Duplicate index definition</p> <p>This error is returned when a duplicate or redundant index is detected during index creation.</p> <p>For example, if you have created an index as follows:</p> <pre>CREATE UNIQUE INDEX IND_1 ON T1(C1,C2,C3);</pre> <p>Next, if you create this index:</p> <pre>CREATE INDEX IND_2 ON T1(C2,C3,C1,C4);</pre>

Error code	Description
	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.
Table Error 13200	Update failed. Used isolation level requires FOR UPDATE.
Table Error 13201	Delete failed. Used isolation level requires FOR UPDATE.
Table Error 13202	Cluster connection does not support isolation levels higher than READ COMMITTED.

D.9 Solid Server Errors

Table D.9. Solid Server Errors

Error code	Description
Server Error 14501	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 (CarrierGrade option) errors. See <i>solidDB High Availability User Guide</i> for details.
Server Error 14502	RPC parameter is invalid. A network error has occurred.
Server Error 14503	Communication error. A communication error has occurred.
Server Error 14504	Duplicate cursor name cursor. You have tried to declare a cursor with a cursor name which is already in use. Use another name.
Server Error 14505	Connect failed, illegal user name or password.

Error code	Description
	You have entered either a user name or a password that is not valid.
Server Error 14506	The server is closed, no new connections allowed. You have tried to connect to a closed server. Connecting was aborted.
Server Error 14507	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.
Server Error 14508	The operation has timed out. You have launched an operation that has been aborted.
Server Error 14509	Version mismatch. A version mismatch has occurred. The client and server are different versions. Use same versions in the client and the server.
Server Error 14510	Communication write operation failed. A write operation failed. This indicates a network problem. Check your network settings.
Server Error 14511	Communication read operation failed. A read operation failed. This indicates a network problem. Check your network settings.
Server Error 14512	There are users logged to the server. You can not shutdown the server now. There are users connected to the server.
Server Error 14513	Backup process is active. You cannot shut down the server now. The backup process is active
Server Error 14514	Checkpoint creation is active. You cannot shut down the server now. The checkpoint creation is active.
Server Error 14515	Invalid user id. You tried to drop a user, but the user id is not logged in to the server.
Server Error 14516	Invalid user name. You tried to drop a user, but the user name is not logged in to the server.

Error code	Description
Server Error 14517	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.
Server Error 14518	Connection to the server is broken, connection lost. Possible network error. Reconnect to the server.
Server Error 14519	The user was thrown out from the server, connection lost. Possible network error.
Server Error 14521	Failed to create a new thread for the client.
Server Error 14529	The operation timed out.
Server Error 14530	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.
Server Error 14531	Too many open cursor, max limit is <i>value</i> . There are too many open cursors for one client; maximum number of open cursors for one connection is 1000. The value can be changed using a configuration value: [Srv] MaxOpenCursors=n
Server Error 14533	Operation cancelled Operation was cancelled because client application called ODBC or JDBC cancel function.
Server Error 14534	Only administrative statements are allowed. Only administrative statements are allowed for the connection.
Server Error 14553	Backup process is not active This error is given if ADMIN COMMAND 'abort backup' is issued and no backup is active.

Error code	Description
Server Error 14554	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.
Server Error 14555	Netbackup: Conflicting usage of backup directory %s.
Server Error 14556	Netbackup: No server connection string specified.
Server Error 14557	Netbackup: A server configured for hot standby cannot act as a netbackup server.
Server Error 14600	Command is ambiguous in cluster session.
Server Error 30150	Server not started This error is given if the solidDB server cannot be started.

D.10 Solid Communication Errors

Table D.10. Solid Communication Errors

Error code	Description
Session Error 20001	Illegal session class.
Session Error 20002	Dynamic link library not found.
Session Error 20003	Wrong dynamic link library version.
Session Error 20004	Illegal address info.
Session Error 20005	Listening address is in use.
Session Error 20006	Server not found.
Session Error 20007	Illegal control parameter.
Session Error 20008	Illegal size parameter.
Session Error 20009	Write operation failed. This error is returned if the server or client is trying to write to an underlying communication channel (socket, named pipe, shared memory, etc.) that is broken.
Session Error 20010	Read operation failed.
Session Error 20011	Accept operation failed.

Error code	Description
Session Error 20012	Network not found.
Session Error 20013	Out of network resources.
Session Error 20023	Too many name resolver requests already in progress.
Session Error 20024	Timeout while resolving host name.
Session Error 20025	Timeout while connecting to a remote host.
Communication Error 21300	Protocol <i>protocol</i> is not supported. Protocol is not supported.
Communication Error 21301	Cannot load the dynamic link library <i>library</i> 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.
Communication Error 21302	Wrong version of dynamic link library <i>library</i> . The version of this library is wrong. Update this library to a newer version.
Communication Error 21303	Network adapter card is missing or needed <i>protocol</i> software is not running. The network adapter card is missing or not functioning.
Communication Error 21304	Out of protocol resources The network protocol is out of resources. Increase the protocols' resources in the operating system.
Communication Error 21305	An empty or incomplete network name was specified. The network name specified is not legal. Check the network name.
Communication Error 21306	Server <i>network name</i> 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.
Communication Error 21307	Invalid connect info <i>network name</i> . The network name given as the connect info is not legal. Check the network name.

Error code	Description
Communication Error 21308	<p>Connection is broken (<i>protocol read/write</i> operation failed with code <i>internal code</i>).</p> <p>The connection using the protocol is broken. Either a read or a write operation has failed with an internal error <i>internal code</i>.</p>
Communication Error 21309	<p>Failed to accept a new client connection, out of <i>protocol</i> resources.</p> <p>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.</p>
Communication Error 21310	<p>Failed to accept a new client connection, listening of <i>network name</i> interrupted.</p> <p>The server was not able to establish a new client connection. The listening has been interrupted.</p>
Communication Error 21311	<p>Failed to start a selecting thread for <i>network name</i>.</p> <p>A thread selection has failed for <i>network name</i>.</p>
Communication Error 21312	<p>Listening info <i>network name</i> already specified for this server.</p> <p>A network name has already been specified for this server. A server can not use a same network name more than once.</p>
Communication Error 21313	<p>Already listening with the network name <i>network name</i>.</p> <p>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.</p>
Communication Error 21314	<p>Cannot start listening, network name <i>network name</i> is used by another process.</p> <p>The server can not start listening with the given network name. Another process in this computer is using the same network name.</p>
Communication Error 21315	<p>Cannot start listening, invalid listening info <i>network name</i>.</p> <p>The server can not start listening with the given listening info. The given network name is invalid. Check the syntax of the network name.</p>
Communication Error 21316	<p>Cannot stop the listening of <i>network name</i>. There are clients connected.</p>

Error code	Description
	You can not stop listening of this network name. There are clients connected to this server using this network name.
Communication Error 21317	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.
Communication Error 21318	Operation failed because of an unusual <i>protocol</i> return code <i>code</i> . Possible network error. Create connection again.
Communication Error 21319	RPC request contained an illegal version number. Either the message was corrupted or there may be a mismatch between server and client versions.
Communication Error 21320	Called RPC service is not supported in the server. There maybe a mismatch between server and client versions.
Communication Error 21321	Protocol <i>protocol</i> is not valid, try using switch '-a' for specifying another adapter id instead of <i>switch</i> . This is returned if the NetBIOS LAN adapter id given in listen/connect string is not valid.
Communication Error 21322	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.
Communication Error 21323	Protocol <i>protocol</i> can not be used for listening in this environment. This message is displayed if the server end communication using specified protocol is not supported.
Communication Error 21324	The process does not have the privilege to create a mailbox.
Communication Error 21325	Only one listening name is supported in this server.
Communication Error 21326	Failed to establish an internal <i>number</i> socket connection code <i>number</i> . solidDB uses one connect socket for internal use. Creation of this socket has failed; the local loopback may not be working correctly.
Communication Error 21327	Too many name resolver requests already in progress.
Communication Error 21328	Timeout while resolving host name.

Error code	Description
Communication Error 21329	Timeout while connecting to host.
RPC Error 21500	Illegal Ping RPC sequence number. A message was either lost or duplicated.
RPC Error 21501	Corrupted Ping message.
RPC Error 21502	Incomplete Ping message. Part of the data was lost.
RPC Error 21503	Extra bytes in Ping message or header corrupted.
RPC Error 21504	Requested Ping level is not currently allowed in server. Start listening with -p%d option.
RPC Error 21505	Illegal Ping buffer size or message corrupted.
RPC Error 21506	Ping session was disconnected abnormally because of a communication error.
RPC Error 21508	Ping feature is not supported in the server. Update your server.
RPC Error 21509	Failed to write to file '%.80s'.
RPC Error 21510	Failed to read from file '%.80s'.

D.11 Solid Communication Warnings

Table D.11. Solid Communication Warnings

Error code	Description
Warning Code 21100	Illegal value <i>value</i> for configuration <i>parameter</i> parameter, using default. An illegal value was given to the <i>parameter</i> parameter. The server will use a default value for this parameter.
Warning Code 21101	Invalid protocol definition <i>protocol</i> in configuration file. The protocol is defined illegally in the configuration file. Check the syntax of the definition.

D.12 Solid Procedure Errors

Table D.12. Solid Procedure Errors

Error code	Description
Procedure Error 23002	Undefined cursor <i>cursor</i> .

Error code	Description
	You have used a cursor that has not been defined in a procedure definition.
Procedure Error 23003	Illegal SQL operation <i>operation</i> .
Procedure Error 23004	Syntax error: parse error, line <i>line number</i> . Check the syntax of your procedure.
Procedure Error 23005	Procedure <i>procedure</i> not found.
Procedure Error 23006	Wrong number of parameters for procedure <i>procedure</i> .
Procedure Error 23007	Procedure name <i>value</i> conflicts with an existing entity. Choose a unique name for a procedure. The specified name is already used.
Procedure Error 23010	Incompatible event <i>event</i> parameter type, line <i>line number</i> .
Procedure Error 23011	Wrong number of parameter for event <i>event</i> , line <i>line number</i> .
Procedure Error 23012	Duplicate wait for event <i>event</i> , line <i>line number</i> .
Procedure Error 23013	Undefined sequence <i>sequence</i> .
Procedure Error 23014	Duplicate sequence name <i>sequence</i> .
Procedure Error 23015	Sequence <i>sequence</i> not found.
Procedure Error 23016	Incompatible variable type in call to sequence <i>sequence</i> , line <i>line number</i> .
Procedure Error 23017	Duplicate symbol <i>symbol</i> . You have duplicate definitions for a symbol.
Procedure Error 23018	Procedure owner <i>owner</i> not found.
Procedure Error 23019	Duplicate cursor name ' <i>cursor</i> '
Procedure Error 23020	Illegal option <i>option</i> for WHENEVER SQLERROR ... statement.
Procedure Error 23021	RETURN ROW not allowed in procedure with no return type, line <i>line number</i> .
Procedure Error 23022	SQL String variable <i>variable</i> must be of character data type, line <i>line number</i> .
Procedure Error 23023	Call syntax error: <i>syntax</i> , line <i>line number</i> .
Procedure Error 23024	Trigger <i>trigger_name</i> not found. Trigger name not found.
Procedure Error 23025	Trigger name <i>trigger_name</i> conflicts with an existing entity.

Error code	Description
	Trigger name conflicts with some other database object. Triggers share the same name space, as for example, in table and procedures.
Procedure Error 23026	Variable <i>variable</i> is of character type, line <i>line number</i> . A CHAR or WCHAR variable is required for the operations like RETURN SQLERROR variable.
Procedure Error 23027	Duplicate reference to column <i>column_name</i> in trigger definition. One column can be referenced only once in the trigger definition.
Procedure Error 23028	Commit and rollback are not allowed in triggers. Trigger body may not contain commit or rollback statements.
Procedure Error 23029	Commit and rollback are not allowed in functions.
Procedure Error 23030	Function <i>function_name</i> not found
Procedure Error 23501	Cursor <i>cursor</i> is not open.
Procedure Error 23502	Illegal number of columns in EXECUTE ... <i>procedure</i> in cursor <i>cursor</i> . You will see this message if the number of columns that you selected does not match the number of variables in the INTO clause.
Procedure Error 23503	Previous SQL operation <i>operation</i> failed in cursor <i>cursor</i> .
Procedure Error 23504	Cursor <i>cursor</i> is not executed.
Procedure Error 23505	Cursor <i>cursor</i> is not a SELECT statement.
Procedure Error 23506	End of table in cursor <i>cursor</i> .
Procedure Error 23508	Illegal assignment, line <i>line number</i> .
Procedure Error 23509	In <i>procedure</i> line <i>line number</i> Stmt <i>statement</i> was not in error state in RETURN SQLERROR OF ...
Procedure Error 23510	In <i>procedure</i> line <i>line number</i> Transaction cannot be set read only, because it has written already.
Procedure Error 23511	In <i>procedure</i> line <i>line number</i> USING part is missing for dynamic parameters for <i>procedure</i> .
Procedure Error 23512	In <i>procedure</i> line <i>line number</i> USING list is too short for <i>procedure</i> .
Procedure Error 23513	In <i>procedure</i> line <i>line number</i> Comparison between incompatible types <i>data type</i> and <i>data type</i> .

Error code	Description
Procedure Error 23514	In <i>procedure</i> line <i>line number</i> type <i>data type</i> is illegal for logical expression.
Procedure Error 23515	<p>In <i>procedure</i> line <i>line number</i> assignment of <i>parameter</i> parameter in list <i>list</i> failed.</p> <p>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 <code>TIMESTAMP</code>, then replace</p> <pre>WHEN ? IS NULL</pre> <p>with</p> <pre>WHEN CAST(? AS TIMESTAMP) IS NULL</pre>
Procedure Error 23516	In <code>CALL procedure</code> , assignment of parameter <i>parameter</i> failed.
Procedure Error 23518	<p>User error: <i>error_text</i></p> <p>User generated error in a procedure or trigger. User can generate this error by using a statement <code>RETURN SQLERROR string</code> or <code>RETURN SQLERROR variable</code>. Variable must be of <code>CHAR</code> or <code>WCHAR</code> type.</p>
Procedure Error 23519	<p>Fetch previous is not supported for procedures.</p> <p>Fetch previous row does not work for result sets returned by a procedure.</p>
Procedure Error 23520	Invalid link name given in remote procedure call.
Procedure Error 23521	Link name not given in remote procedure call.
Procedure Error 23522	Dynamic parameters not allowed with remote procedure call.
Procedure Error 23523	Default node not defined.
Procedure Error 23524	Could not load application.
Procedure Error 23525	Function not found from the DLL.
Procedure Error 23526	In <code>CALL <procedure_name></code> assignment of default value of parameter <code><parameter_number></code> failed.

Error code	Description
	This error message occurs if you call a procedure with too few parameters and you have not specified default values for the missing parameters.
Procedure Error 23527	In CALL <procedure_name> parameter <parameter_number> assigned twice. This occurs if you specify the same parameter more than once.
Procedure Error 23528	Application is already running.
Procedure Error 23529	Application is not running.

D.13 Solid Sorter Errors

Table D.13. Solid Sorter Errors

Error Code	Meaning
Sorter Error 24001	Sort failed due to insufficient configured TmpDir space
Sorter Error 24002	Sort failed due to insufficient physical TmpDir space
Sorter Error 24003	Sort failed due to insufficient sort buffer space
Sorter Error 24004	Sort failed due to too long row (internal failure)
Sorter Error 24005	Sort failed due to I/O error
Sorter Error 30802	Failed to create a temporary file for local sorting (system errno =) The sorter cannot create a temporary file.
Sorter Error 30803	Illegal value specified for parameter: [%s]%s=%u(legal range is %u-%u)
Sorter Error 30804	Sorter temporary directory: %s does not exist

D.14 Solid SpeedLoader Utility (solload) Errors

Table D.14. Solid SpeedLoader Utility (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.

Error Code	Meaning
106	Illegal column name This error applies to the column name used in the control file.
107	Illegal constraint
108	Invalid column data The data type in the data file conflicts with the table definition.
109	Unique constraint violation
110	Concurrency conflict, two transactions updated or deleted the same row
112	Unsupported character set
114	Null data in NOT NULL column NULL data value given in a NOT NULL column
116	Communication error, connection is lost
121	RPC parameter error
122	Table not found
124	Wrong number of parameters

Appendix E. solidDB ADMIN COMMAND Syntax

This appendix describes the solidDB ADMIN COMMAND syntax. This command set is not part of ANSI SQL; it is a solidDB-specific extension.

E.1 ADMIN COMMAND

```
ADMIN COMMAND 'command_name'
```

```
command_name ::= ABORT | ASSETEXIT | BACKUP |  
BACKUPLIST | BACKUPSERVERON | CHECKPOINTING | CLEANBGJOBINFO |  
CLOSE | DESCRIBE | ERRORCODE | ERROREXIT | EXIT |  
FILESPEC | HELP | HOTSTANDBY | INFO | MAKECP | MEMORY | MESSAGES  
| MONITOR | NETBACKUP | NETBACKUPLIST | NOTIFY | OPEN | OPT |  
PARAMETER | PERFMON | PID | PINGTEST | PROCTRACE |  
PROTOCOLS | REPORT | RUNMERGE | SAVE | SETREADONLYFLAG | SHUTDOWN  
| SQLLIST | STARTMERGE | STATUS | THRINFO | THROWOUT | TRACE |  
USERID | USERLIST | USERTRACE | VERSION
```

E.1.1 Supported in

ADMIN COMMAND syntax is supported in all solidDB editions.

E.1.2 Usage

This SQL extension executes administrative commands. The `command_name` in the syntax is a SolidConsole or Solid SQL Editor (solsql) command string, for example:

```
ADMIN COMMAND 'backup'
```

If you are entering these commands using Solid Remote Control (solcon), be sure to specify the syntax with command name only (without the quotes), for example:

```
backup
```

Abbreviations for ADMIN COMMANDs are also available, for example,

```
ADMIN COMMAND 'bak'
```

To access a list of abbreviated commands, execute

```
ADMIN COMMAND 'help'
```

The result set contains two columns: RC INTEGER and TEXT VARCHAR(254). Integer column RC is a command return code (0 if success), and varchar column TEXT is the command reply. The TEXT field contains the same lines that are displayed on SolidConsole screen, one line per one result row.

Note that all options of the ADMIN COMMAND are not transactional and cannot be rolled back.



Caution

ADMIN COMMANDS and Starting Transactions

Although ADMIN COMMANDs are not transactional, they will start a new transaction if one is not already open. (They do not commit or roll back any open transaction.) This effect is usually insignificant. However, it may affect the 'start time' of a transaction, and that may occasionally have unexpected effects. solidDB's concurrency control is based on a versioning system; you see a database as it was at the time that your transaction started. (See the section of *solidDB Administration Guide* titled 'Solid Bonsai Tree Multiversioning and Concurrency Control'). So, for example, if you: commit work, and issue an ADMIN COMMAND without doing another commit, and go to lunch and return an hour later, then your next SQL command may see the database as it was an hour ago, i.e. when you first started the transaction with the ADMIN COMMAND.



Caution

Error codes in ADMIN COMMANDS ADMIN COMMANDS return an error only if the command syntax or parameter values are incorrect. That is, 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: TC and TEXT. The RC (return code) column contains the return code of the operation: it is "0" for success, and different numeric values for errors. It is thus necessary to check both the codes (of the ADMIN COMMAND statement and of the operation).

Following is a description of the syntax for each ADMIN COMMAND command option:

Table E.1. ADMIN COMMAND Syntax

Option Syntax	Description
ADMIN COMMAND 'abort [backup netbackup]'	<p>Aborts the active local or network backup process. The backup operation is not guaranteed to be atomic, therefore the cancelled operation may produce an incomplete backup file to the backup directory until the next backup takes place.</p> <p>If the options are not entered, the default behaviour is similar to command ADMIN COMMAND 'abort backup'.</p>
ADMIN COMMAND 'assertexit' Abbreviation: asex	Asserts the server.
ADMIN COMMAND 'backup [-s] [backup_directory]' Abbreviation: bak	<p>Makes a backup of the database. The operation can be performed as a synchronized or an asynchronic (default) manner. The synchronized operation is specified by using the optional -s parameter.</p> <p>The default backup directory is the one defined in the <i>[General]</i> section of the configuration parameter <i>BackupDirectory</i>. The backup directory may also be given as an argument. For example, backup abc creates a backup in directory 'abc'. All directory definitions are relative to the solidDB working directory.</p>
ADMIN COMMAND 'backuplist' Abbreviation: bls	Displays a status list of last local backups.
ADMIN COMMAND 'backupserveron' Abbreviation: bakson	Sets the server to backupserver mode.
ADMIN COMMAND 'cleanbgjobinfo' Abbreviation: cleanbgi	Cleans the table SYS_BACKGROUNDJOB_INFO containing status data of background procedures.

Option Syntax	Description
ADMIN COMMAND 'checkpointing' Abbreviation: cp	Turns on/off checkpointing.
ADMIN COMMAND 'close' Abbreviation: clo	Closes the server to new connections; no new connections are allowed.
ADMIN COMMAND 'describe parameter <i>param</i> ' Abbreviation: des	<p>Returns a description of the specified parameter.</p> <p>Note that the param should be in the form <i>section_name.param_name</i>. The section and parameter names are case-insensitive.</p> <p>The following example describes parameter <i>Com.Trace</i> = <i>y/n</i></p> <p>ADMIN COMMAND 'des parameter com.trace'</p>
ADMIN COMMAND 'errorcode {all <i>SOLID_error_code</i> }' Abbreviation: ec	Displays a description of an error code (or all codes). Give the code number as an argument, for example, errorcode 10033 .
ADMIN COMMAND 'errorexit <number>' Abbreviation: erex	Forces the server into an immediate process exit with the given process exit code.
ADMIN COMMAND 'filespec' Abbreviation: fs	Displays database file specifications, current fill ratios and current file sizes.

Option Syntax	Description
ADMIN COMMAND 'help' Abbreviation: ?	Displays available commands.
ADMIN COMMAND 'hotstandby [option]' Abbreviation: hsb	A HotStandby command. For list of options see <i>solidDB High Availability User Guide</i> .
ADMIN COMMAND 'info options' Abbreviation: info	Returns server information. Options are one or more of the following values, each separated by a space: <ul style="list-style-type: none">• numusers - Number of current users.• sernum - Server serial number.• dbsize - Database size.• dbfreesize - Amount of free space remaining in database.• 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.• logsize - Size of log files.• uptime - Server up since.• bcktime - Timestamp of last successfully completed local backup.• netbackuptime - Timestamp of last successfully completed network backup.• cptime - Timestamp of last successfully completed checkpoint.• tracestate - Current trace state.

Option Syntax	Description
	<ul style="list-style-type: none">monitorstate - Current monitor state, which is the number of users who have SQL monitoring currently enabled; this value is -1 if all users have SQL monitoring enabled. Note that SQL monitoring is enabled using the ADMIN COMMAND 'monitor {on off} [user {username userid}]' (described below).openstate - Current open or close state — that is, whether the database server accepts new connections or not. "open" means that the database server accepts new connections.primarystarttime - The time the Primary role has started.secondarystarttime- The time the Secondary role has started. <p>More than one option can be used per command. Values are returned in the same order as requested, one row for each value.</p> <p>Example command:</p> <p>ADMIN COMMAND 'info dbsize logsize'</p> <p>Example output:</p> <pre>RC TEXT 0 851968 0 573440</pre>
ADMIN COMMAND 'makecp [-s]' Abbreviation: mcp	<p>Makes a checkpoint. Requires SYS_ADMIN_ROLE privilege.</p> <p>By default, the checkpoint is asynchronous. With the option -s, the command returns only after the checkpoint has completed.</p>

Option Syntax	Description
ADMIN COMMAND 'memory' Abbreviation: mem	Returns the server process memory size. The reported process memory size can differ from the process size reported by your operating system.
ADMIN COMMAND 'messages [{ warnings errors}] [count]' Abbreviation: mes	Displays server messages. Optional severity and message numbers can also be defined. For example: ADMIN COMMAND 'messages warnings 100' displays last 100 warnings.
ADMIN COMMAND 'monitor {on off} [user {username userid}]' Abbreviation: mon	Sets server monitoring on and off. Monitoring logs user activity and SQL calls to <code>soltrace.out</code> file.
ADMIN COMMAND 'netbackup [options] [DELETE_LOGS KEEP_LOGS] [connect connect str] [dir backup dir]' Abbreviation: nbak	<p>Makes a network backup of the database. The operation can be performed as a synchronized or an asynchronic (default) manner. The synchronized operation is specified by using the optional <code>-s</code> parameter.</p> <p>If you use the <code>DELETE_LOGS</code> parameter, backed-up log files in the source server are deleted. This is sometimes referred to as Full backup. This is the default value. On the other hand, if you use the <code>KEEP_LOGS</code> parameter, backed-up log files are kept in the source server. This is sometimes referred to as Copy backup. Using the keyword <code>KEEP_LOGS</code> corresponds to setting the <i>General</i> parameter <i>Netbackup-DeleteLog</i> to "no".</p> <p>The default connect string and the default netbackup directory are defined in the <i>NetBackupConnect</i> and in the <i>Net-BackupDirectory</i> parameters in the <i>[General]</i> section of the configuration file.</p> <p>Options that are entered with the netbackup command override the values specified in the configuration file. Directory definitions are relative to the solidDB working directory.</p>

Option Syntax	Description
ADMIN COMMAND 'netbackuplist' Abbreviation: nbls	Displays a status list of the most recently made network backups of the database server.
ADMIN COMMAND 'notify user {username user id ALL } message' Abbreviation: not	<p>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.</p> <p>The following example sends a notify message to a user with user id 5 ; the event then gets the value of the message parameter.</p> <p>ADMIN COMMAND 'notify user 5 Canceled by admin'</p>
ADMIN COMMAND 'open' Abbreviation: ope	Opens server for new connections; new connections are allowed.
ADMIN COMMAND 'opt accelerator diskless hsb purify sync'	Displays whether the requested option is enabled or disabled.
ADMIN COMMAND 'parameter [option][name[= [* value][temporary]]' Abbreviation: par	<p>Displays and sets server parameter values. If you run the command without a specified value, the parameter will be set to its startup value. If you assign a parameter value with an asterisk (*), the parameter will be set to its factory value. The "name" may be either a section name, or it may be a parameter name prefaced by a section name and period (e.g. "com.trace"). For example:</p> <ul style="list-style-type: none"> <i>parameter</i> used alone displays all parameters.

Option Syntax	Description
	<ul style="list-style-type: none"><code>parameter general</code> displays all parameters from section <code>[General]</code>.<code>parameter general.readonly</code> displays a single parameter named <code>readonly</code> from section <code>[General]</code>. You must place a period between the section name (<code>[General]</code>) and the parameter name (<code>readonly</code>).<code>parameter com.trace=yes</code> sets communication trace on. You must place a period between the section name (e.g. <code>[Com]</code>) and the parameter name (e.g. <code>trace</code>). You should not put blanks around the equals sign.<code>parameter com.trace=</code> sets communication trace to its startup value.<code>parameter com.trace=*</code> sets communication trace to its factory value. <p>The output may contain three values, as shown below:</p> <pre>0 Logging DurabilityLevel 1 2 3</pre> <p>The three values represent the following:</p> <ul style="list-style-type: none">1 is the current value (may be set dynamically)2 is the value in the INI file (startup value)3 is the factory value <p>If the <code>-r</code> option is used, then only the current parameter values are returned.</p>
<pre>ADMIN COMMAND 'perfmon [- c - r] [options] [diff [start stop] [filename interval]</pre>	<p>Returns server performance counters. The options are:</p> <ul style="list-style-type: none"><code>-c</code> - prints actual counter values. If this option is not provided, the output numbers are operations/second where applicable.

Option Syntax	Description
<p>[<i>name_prefix_list</i>]'</p> <p>Abbreviation: pmon</p>	<ul style="list-style-type: none"> • -r - prints in raw mode, which includes only the latest counter values without any formatting. No option names or additional information is printed. This option is useful if actual monitoring is performed using some other external program that retrieves the counter values from the server. • -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 perfmon execution instead of the absolute value • diff - starts a server task that prints out all perfmon counters with specified intervals to a file. The interval must be given in milliseconds. The output file is written using "comma-separated values", with the first row including counter names. The file, as it is, can be processed by spreadsheet programs like Excel. • <i>name_prefix_list</i> - limits output to specific counter names. For example, to print all file related counters, the <i>name_prefix_list</i> should be file. You can also specify multiple prefixes. <p>The following example returns all information:</p> <p>ADMIN COMMAND 'perfmon'</p> <p>The following example returns all values whose name starts with prefix file and cache as counters.</p> <p>ADMIN COMMAND 'perfmon-c file cache'</p> <p>Note that the prefix file and cache are matched to those counter names that are in the perfmon output.</p>

Option Syntax	Description
	<p>The following example starts a diff task that writes to myd.csv file on 1000 milliseconds interval:</p> <p>ADMIN COMMAND 'pmon diff start myd.csv 1000'</p> <p>For sample output, and description of counters, see the section of <i>solidDB Administration Guide</i> titled "Detailed DBMS Monitoring and Troubleshooting".</p>
<p>ADMIN COMMAND 'pid'</p> <p>Abbreviation: pid</p>	<p>Returns server process id.</p>
<p>ADMIN COMMAND 'pingtest <servername><level>'</p>	<p>Performs an asynchronous ping test.</p> <p>For more information on Ping facility levels, see <i>solidDB Administration Guide</i>, chapter "The Ping Facility".</p>
<p>ADMIN COMMAND 'proctrace { on off } user <i>username</i> { procedure trigger table } <i>entity_name</i>'</p> <p>Abbreviation: ptrc</p>	<p>This turns on tracing in stored procedures and triggers.</p> <p>The "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 SmartFlow, 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.</p> <p>The "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.</p> <p>For more detail about proctrace, see "Tracing Facilities For Stored Procedures And Triggers" in <i>solidDB SQL Guide</i>.</p> <p>See also the discussion of usertrace on page D-15.</p>

Option Syntax	Description
ADMIN COMMAND 'protocols' Abbreviation: prot	Returns a list of available communication protocols, one row for each protocol. Example: ADMIN COMMAND 'protocols'
ADMIN COMMAND 'report <i>filename</i> ' Abbreviation: rep	Generates a report of server information to a file given as an argument.
ADMIN COMMAND 'runmerge' Abbreviation: rm	Runs an index merge.
ADMIN COMMAND 'save parameters [<i>filename</i>]' Abbreviation: save	Saves the set of current configuration parameter values to a file. If no file name is given, the default <code>solid.ini</code> file is rewritten. This operation is performed implicitly at each checkpoint.
ADMIN COMMAND 'setreadonlyflag {yes no}' Abbreviation: srof	Sets the read-only flag on and off.
ADMIN COMMAND 'shutdown [<i>force</i>]' Abbreviation: sd	Stops solidDB. If the "force" option is used, the active transactions are aborted and the users are disconnected forcefully.
ADMIN COMMAND 'sqlist top <i>number_of_statements</i> '	This command prints out a list of the longest running SQL statements among the currently running statements. The list contains the selected number of statements.

Option Syntax	Description
ADMIN COMMAND 'status' Abbreviation: sta	Displays server statistics.
ADMIN COMMAND 'status backup netbackup' Abbreviation: sta backup netbackup	<p>Displays status of the last started local or network backup. The status can be one of the following:</p> <ul style="list-style-type: none"> • 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 last backup failed, the output is: <i>errorcode</i> ERROR where the <i>errorcode</i> shows the reason for the failure
ADMIN COMMAND 'startmerge' Abbreviation: sm	Starts and waits for completion of merge.
ADMIN COMMAND 'thrinfo' Abbreviation: ti	Displays thread related information.
ADMIN COMMAND 'throwout {username userid all}' Abbreviation: to	Exits users from solidDB. To exit a specified user, give the user id as an argument. To throw out all users, use the keyword ALL as an argument.
ADMIN COMMAND 'trace { on off} sql rpc sync info <level>	<p>Sets server trace on or off. The tracing options are:</p> <ul style="list-style-type: none"> • sql - SQL messages • rpc - Network communications

Option Syntax	Description
<pre>flowplans all'</pre> <p>Abbreviation: tra</p>	<ul style="list-style-type: none"> • sync - Synchronization messages • info <level> - SQL execution trace (level is 0...8) • flowplans - plans of Flow SQL statements <p>If no options are specified, or all is specified, both SQL messages and network communications messages are written to the trace file. The name of the default trace file is <code>sol-trace.out</code>.</p>
<pre>ADMIN COMMAND 'userid'</pre> <p>Abbreviation: uid</p>	<p>Returns the user identification number of the current connection.</p> <p>Example:</p> <pre>ADMIN COMMAND 'userid'</pre>
<pre>ADMIN COMMAND 'userlist [-l]</pre> <pre>[name id]'</pre> <p>Abbreviation: ul</p>	<p>This command displays a list of users currently logged in to the database, together with a number of primary attributes. These attributes are: User name, User Id, Type, Machine Id, Login time and Appinfo (optional). For attribute descriptions, see the detailed output description below.</p> <p>Option -l (long) displays a more detailed output. The fields in the long output are:</p> <ul style="list-style-type: none"> • <i>Id</i> - The user session identification number within the database. The lifetime of an Id is that of the user session. After a user logs out, the number may be reused. • Type - Client type. Possible values are: <ul style="list-style-type: none"> • Java, which refers to a client using JDBC, such as the SolidConsole • ODBC, which refers to a client using ODBC • SQL, which refers to Solid's SolSql editor

Option Syntax	Description
	<ul style="list-style-type: none"> • <i>Machine</i> - The client computer name (host name) and its IP address, if available. • <i>Appinfo</i> - The value of the client computer's environmental variable <i>SOLAPPINFO</i>, if the client is using ODBC. In the case of JDBC, the Java utility property <i>solid_appinfo</i> has to be set to that value, for it to be visible in the output. 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 <p>Note: the value of SOLAPPINFO must not contain blanks.</p> • <i>Autocommit</i> - If the autocommit mode 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 (value 1), each statement is automatically committed. • <i>Last activity</i> - The time when the client last time sent a request to the server. • <i>RPC compression</i> - Indicates whether the data transmission compression is on or off. • <i>Transparent failover</i> - This field indicates if Transparent Failover (TF) is in use. Transparent failover is a characteristic of the CarrierGrade configuration. It hides the server role change from the user. Because Solid Tools do not support TF, you will only see a no value in this field. • <i>Transaction active</i> - 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.

Option Syntax	Description
	<ul style="list-style-type: none"> • <i>Transaction duration</i> - This field indicates the duration of the currently open transaction. After COMMIT or ROLLBACK, the value becomes 0. • <i>Transaction isolation</i> - 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. • <i>RPC seqno</i> - Internal protocol message sequence number. • <i>SQL sortarray</i> - The size of user-specific internal sort array. • <i>SQL unionsfromors</i> - The value tells how many (at most) OR operators may be converted to UNIONs. Unions are faster but require more memory to execute. • <i>Stmt id</i> - The current statement identification number. The numbers are session specific and they are assigned for each different statement • <i>Stmt state</i> - An internal statement execution state. • <i>Stmt rowcount</i> - The number of rows retrieved or inserted in the current statement. • <i>Stmt starttime</i> - The current statement start date and time. • <i>Stmt duration</i> - 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. • <i>Stmt SQL str</i> - The current statement string.
ADMIN COMMAND 'usertrace { on off } user <i>username</i>	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.

Option Syntax	Description
<pre>{ procedure trigger table } entity_name' Abbreviation: utrc</pre>	<p>The "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 SmartFlow, 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.</p> <p>The "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, then it will generate output for all triggers on that table. Trace is activated only when the specified user calls the procedure / trigger.</p> <p>For more detail about proctrace, see "Tracing Facilities For Stored Procedures And Triggers" in solidDB SQL Guide.</p> <p>See also the discussion of "proctrace" on page D-10.</p>
<pre>ADMIN COMMAND 'version' Abbreviation: ver</pre>	<p>Displays server version info.</p>

Glossary

This glossary gives you a description of the terminology used in this guide.

A

ACID

ACID is short for Atomicity - Consistency - Isolation - Durability and describes the four properties of an enterprise-level transaction:

- **ATOMICITY:** a transaction must be done or undone completely. In the event of an error or failure, all data manipulations must be undone, and all data must be rolled back to its previous state.
- **CONSISTENCY:** a transaction must transform a system from one consistent state to another consistent state.
- **ISOLATION:** each transaction must happen independently of other transactions occurring at the same time.
- **DURABILITY:** Completed transactions must remain stable/permanent, even during system failure.

B

Binary Large Object (BLOB)

"BLOB" is an acronym for Binary Large OBject. A BLOB is a large block of information such as a picture, video clip, sound excerpt, or a document that contains any non-printable formatting characters.

BLOB information is usually stored in a high capacity, variable-length binary data type. With solidDB database servers, BLOB data is usually stored in VARBINARY. However, this is not always necessary. Although BLOBs are generally Binary and Large, and are usually stored in variable-length data types, none of these characteristics are required. Depending upon the actual data value, you might store your data in a fixed-length BINARY field rather than a variable-length VARBINARY field. If your data is composed entirely of standard characters, then you might store the data in one of the various high-capacity character data types, such as VARCHAR. (BLOBs that are composed entirely of printable characters are sometimes called CLOBs. Since BINARY fields can store any data that CHAR fields can store, CLOBs can be stored in either CHAR or BINARY fields. CLOBs are a subset of BLOBs.)

For a complete list of the BINARY and CHAR data types supported by solidDB, see "Binary Data Types" and "Character Data Types" in solidDB SQL Guide.

Note that if you are using in-memory tables, BLOB lengths are restricted to approximately the size of the page. See the appendices in *solidDB In-Memory Database User Guide* for an explanation of how to calculate the approximate maximum size of a BLOB in an in-memory table.

With the exception of the in-memory table restriction listed above, solidDB generally treats BLOB/CLOB the same way as any other BINARY/CHAR data. You do not need to do anything special to store or retrieve such data.

C

Cache

solidDB, like almost any database server, does not perform database operations (insert/update/delete) directly on the disk. Instead, it keeps some of the most recently used data in memory. This data, along with other information, is stored in the server's "cache". Most of this data is stored in "pages" that correspond to "pages" of the database file that is stored on the disk drive.

The size of the cache is determined by a `solid.ini` configuration parameter named *CacheSize*.

Note that "cache" is not the same as the CPU cache that exists at the hardware level.

Catalog

A catalog logically partitions a solidDB database so that data is organized in ways that meet business or application requirements. Each logical database is a catalog and contains a complete, independent group of database objects, such as tables, indexes, procedures, triggers, etc. Note, however, that a solidDB catalog contains a variety of data objects, not just indexes (as in the traditional sense of a library card catalog, which serves to locate an item without containing the full contents of the item).

Each of these catalogs can act as an independent master or replica database. This makes it possible, for example, to create two or more independent replica databases in one physical local database. It is also possible to have one or more catalogs in this same local database that represent master database(s).

A catalog is also referred to as a node when the catalog has been defined as a master or replica using the SET SYNC NODE command. Each catalog of a solidDB environment must have a node name that is unique within the domain. Assigning the node name is part of the registration process of a replica database.

A catalog can qualify one or more schemas. A schema is a persistent database object that provides a definition for the entire database; it represents a collection of database objects associated with that specific schema name. The catalog name is used to qualify a database object name, such as tables, views, indexes, stored procedures, triggers, and sequences. They are qualified as: *catalog_name.schema_name.database_object* or *catalog_name.user_id.database_object*.

Inside each catalog there may be multiple schemas. It is legal to use the same schema name in more than one catalog. Typically, each user in a catalog is allowed to have his or her own schema(s). Providing users with their own schema allows each user to have his or her own tables (or other database objects) without naming overlaps.

Character Large Object (CLOB)

CLOBs are really a subset of BLOBs. For information about BLOBs, see BLOB in the glossary and index.

See also BLOB.

Checkpoint

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

Note that checkpoints apply to persistent in-memory tables, not just disk-based tables.

In between checkpoints, the server writes committed transactions to a transaction log. If the server fails, any transactions committed since the last checkpoint can be recovered from this transaction log. See also "transaction log".

For more details about checkpoints, see Section 3.11, "Creating Checkpoints" and Section 6.6, "Tuning Checkpoints".

Client/server computing

Client/server computing divides a large piece of software into modules that need not all be executed within the same memory space nor on the same processor. The calling module becomes the 'client' that requests services, and the called module becomes the 'server' that provides services. Client and server processes exchange information by sending messages through a computer network. They may run on different hardware and software platforms as appropriate for their special functions.

Two basic client/server architecture types are called two-tier and three-tier application architectures.

Column

See Table.

Communication Protocol

A communication protocol is a set of rules and conventions used in the communication between servers and clients. The server and client have to use the same communication protocol in order to establish a connection. TCP/IP is an example of a commonly-used communication protocol.

Cursor

Also known as the Current Set Of Records in some database languages. The cursor is a database object pointing to a currently selected set of records.

D

Database Administrator

The database administrator is a person responsible for tasks such as:

- managing users, tables, and indices
- backing up data
- allocating disk space for the database files

Data Definition Language (DDL)

There are two major components of the Structured Query Language (SQL): Data Definition Language (DDL) and Data Manipulation Language (DML). Data Definition Language is used to insert, retrieve and modify data stored within a relational databases. Some of the major commands comprising DDL are CREATE TABLE, DROP TABLE and CREATE INDEX.

Data Manipulation Language (DML)

There are two major components of the Structured Query Language (SQL): Data Definition Language (DDL) and Data Manipulation Language (DML). Data Manipulation Language (DML) is used to insert, retrieve and modify data stored within a relational databases. The major commands comprising DML are SELECT, INSERT, DELETE and UPDATE.

Database Management System (DBMS)

A DBMS is a system that stores information in and retrieves information from a database. A DBMS typically consists of a database server, administration utilities, an application interface, and development tools.

Database Procedures (Stored Procedures)

See stored procedures.

Durability

Durability is a characteristic of a transaction. A transaction is durable if it is recoverable when there has been a failure after a transaction commit. To ensure durability, solidDB servers write transaction data to a log file when the transaction is committed.

See also "Strict Durability" and "Relaxed Durability".

E

Event Alerts

Event alerts are database objects with a name and parameters. Event alerts are used to signal an event in the database. Events allow different applications to coordinate with each other. Events are not sent directly from one application to another. Instead, the sender calls a stored procedure that executes the POST EVENT command, and the receiving application calls a stored procedure that waits on the event.

The use of event alerts removes resource-consuming database polling from applications.

Field

See Table.

I

Incremental Publication

See *solidDB SmartFlow Data Replication Guide*.

Index

An index of records has an entry for each key field (for example, employee name, identification number, etc.) and the location of the record. Indexes are used to speed up access to tables. The database engine uses indexes to access the rows in a table directly. Without indexes, the engine would have to search the whole contents of a table to find the desired row. A single table can have more than one index; however, adding indexes does slow down write operations, such as inserts deletes, and updates on that table. There are two kinds of indexes: non unique indexes and unique indexes. A unique index is an index where all key values are unique.

Intelligent Join Constraint Transfer

In algebra, we know that if

$$x = 3$$

and

$$y = x$$

then

$$y = 3$$

Similarly, in SQL we know that if our WHERE clause says

```
table1.col1 = 3
```

```
and
```

```
table2.col1 = table1.col1
```

```
then for all valid results
```

```
table2.col1 = 3
```

Thus the following queries are equivalent, i.e. they return the same result set:

```
... WHERE table1.col1 = 3 and table2.col1 =  
      table1.col1;
```

```
... WHERE table2.col1 = 3 and table2.col1 =  
      table1.col1;
```

Depending upon the distribution of the data, the clause

```
tableX.col1 = 3
```

may be more selective (i.e. return a smaller percentage of rows) in one table than the other. Thus by "transferring" part of the constraint from one table to the other, and by reordering the join, we may get higher performance.

This optimization technique is called "intelligent join constraint transfer", and it is one of the techniques that the solidDB optimizer uses whenever possible.

Intelligent Transaction

A Solid Intelligent Transaction is an extension to the traditional transaction model. It is a collection of SQL statements that may contain business logic that is typically implemented as Solid stored procedures. These procedures are able to communicate with each other using the Parameter Bulletin Board of the transaction. A transaction that is intelligent is capable of validating itself in the current database and adapting its contents (if required) according to the rules of the transaction.

Since an intelligent transaction is created in the replica database, but is finally committed in the master database, it is a long-lived transaction. Therefore all validity checking of the transaction must be done by the transaction itself.

Isolation Level

See Transaction Isolation Level.

L

Local Data

Data is considered "local" to a database if that data is not shared with any other database. This means the local data is not visible from any other database. In other words, data is local if the data is neither part of a "replica" of another database nor part of a "master" for another database.

See also Local Database.

Local Database

In this guide when discussing a specific example code or an SQL command, the local database refers to the database on which the sample code is running (that is, the database to which a user is connected).

In those places where synchronization is discussed in this guide, it is assumed that the user is connected to the "replica", not the "master"; thus the "replica" database and the "local" database refer to the same database in most examples used in this guide.

In those scenarios where there are three or more levels in a synchronization configuration, the "middle" level may be both a replica of one database and a master to another database that is at a lower level. Again, however, the "local" database, in general, refers to the database to which the user is connected and on which the user executes commands.

See also Local Data.

Lock

Database management systems use locks to facilitate concurrency control. Locks enable different users to access different records or tables within the same database without interfering with one another. Locking mechanisms can be enforced at the record or table levels.

Log File

See Transaction Log File.

M

Master Database (also Known as "master")

See *solidDB SmartFlow Data Replication Guide*.

See also Local Database.

Messages

See *solidDB SmartFlow Data Replication Guide*.

See also Local Database.

N

Network Name

The network name of a server consists of a communication protocol and a server name. This combination identifies the server in the network.

Solid Clients support Logical Data Source Names. These names can be used to give a database a descriptive name. This name is mapped to a network name using either parameter settings in the clients `solid.ini` file or in Microsoft Windows operating systems' registry settings.

Node

See Catalog and Local Database.

O

Open Database Connectivity (ODBC)

ODBC is a programming interface standard for SQL database programs. solidDB offers a native ODBC programming interface.

Optimizer Hints

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

P

Phantom

Database users sometimes refer to "phantom reads" or "phantom updates". A phantom occurs if a record seems to appear partway through a transaction, or appears and disappears within the same transaction. Phantoms can be prevented by using the `SERIALIZABLE` transaction isolation level.

For an example of a situation in which you might get phantoms, suppose that your isolation level is `READ UNCOMMITTED`. During your transaction, you execute the same `SELECT` statement twice (with some other statements in between the two `SELECT`s). As a result of each `SELECT`, you will get all records whose inserts/updates have been committed by other users. But you would not necessarily see the same

records each time because after your first **SELECT** and before your second **SELECT** another user may commit some records that meet the criteria in the **WHERE** clause of your **SELECT** statement.

Publication

See *solidDB SmartFlow Data Replication Guide*.

Q

Query

A query is the primary mechanism for retrieving information from a database. Queries consist of questions presented to the database in a predefined format. solidDB uses the Structured Query Language (SQL) standard query format and a selection of enhancements to it.

R

Record

A record is a complete set of information in a database. Records are composed of different columns in a table and each record is represented with a separate row in this table.

Refresh

When a SmartFlow replica requests data from a master using the command **MESSAGE APPEND RE-FRESH**, the operation is called a "refresh". Note that although the word "refresh" implies that the user has gotten the data at least once before (i.e. this is the 2nd or later request) we use the term loosely to apply to all requests, including the initial one.

Relational Database Management System (RDBMS)

solidDB is an RDBMS, which stores and retrieves information that is organized into two-dimensional tables. This name derives from the relational theory that formalizes the data manipulation requests as set operations and allows mathematical analysis of these sets. RDBMSs typically support the SQL language for data manipulation requests.

Replica Database (also Known as "replica")

A solidDB database that contains a subset of master data and some tentative local transaction data.

See also Local Database.

Relaxed Durability

A transaction has relaxed durability when it becomes durable some time following execution of **Commit**. (The time delay is usually in the range of tens or hundreds of milliseconds, but may be any length.) In this situation, it is possible for data to be lost even though it has been committed. If the server goes down after the commit but before the data is made durable (e.g. written to a log file.), then the data may be lost.

Contrast this with "strict durability", which guarantees that the user is not told that the data was committed until that data has been made durable.

See also "Durability", "Strict Durability".

Row

See Table.

S

Schema

A schema is a database object that may contain other database objects (such as tables, views, etc.); schemas allow you to organize your database objects and schemas prevent multiple users from conflicting when they choose identical object names (such as table names). Within each schema, each data object (such as a table) must have a unique name. However two different users may use the same table name in different schemas, for example, Sue Lamm and Dan Wong could each have a table named table1.

In this way, schemas are like the directories or operating systems. Each directory contains zero or more files; within each directory, each filename must be unique, but two different directories might contain different files with the same name. Schemas are part of a hierarchy in this order: database, database catalog, schema, database object (for example, table).

Within each database, each catalog name must be unique. Within each catalog, each schema name must be unique. Within each schema, each database object name must be unique. Note that a schema cannot contain another schema; in this way schemas are unlike directories. (A directory may contain another directory, but a schema may not contain another schema).

Any table, view, etc. within a database can be uniquely identified by specifying its "fully qualified" name, which includes the catalog name, the schema name, and the database object name, for example:

`sales_catalog.sue_lamm.table1`

`sales_catalog.dan_wong.table1`

Fully-qualified names are always unique. Note also that each table or other database object belongs to exactly one schema; a table may not be part of more than one schema (or more than one catalog).

Schemas do not provide any privacy or security. By specifying the fully qualified name of a database object (such as a table), you may access database objects in other users' schemas (assuming that you have appropriate privileges on those objects); schemas do not prevent you from accessing data owned by other users, or vice versa.

By default, each user has his or her own schema, the name of which is the same as the user's login name. For example, if Sue Lamm logs in as `sue_lamm`, then when she connects to a database she will automatically be connected to the `sue_lamm` schema. She may change to a different schema by using the `SET SCHEMA` command. Within a particular catalog and schema, you do not need to specify the fully-qualified name; for example, if you have already executed:

```
SET CATALOG 'sales_catalog';
```

```
SET SCHEMA 'sue_lamm';
```

then if you specify only `table1`, the database server knows to use the `table1` in the `sue_lamm` schema of the catalog named `sales_catalog`. Although each user has a default schema name based on his or her login, a user is not restricted to owning only that one schema. A user may create additional schemas by using the `CREATE SCHEMA` command.

See also [Catalog](#).

Sequence Objects

Sequence objects generate number sequences for objects stored in databases. Sequences have an advantage over separate tables. They are specifically fine-tuned for fast execution and result in less overhead than normal update statements.

Server Name

When you specify the value of the `solid.ini` configuration file, you must specify the network name of the server. The network name of a server consists of a communication protocol and a server name. This combination identifies the server in the network. The protocol must be one of the standard communication protocols, such as TCP/IP ("`tcp`"), named pipes ("`nmpipe`"), etc. The valid values for the server name depend upon the protocol and on whether the client and server are running on the same computer. The server name might be a name, such as "`calvin`" or "`chicago_office`", or it might be a node name and a service port, such as "`hobbes 1313`", or it might be just a service port, such as "`1313`".

SmartFlow Structured Query Language (SQL) Commands

See *solidDB SmartFlow Data Replication Guide*.

SQL Access Group's Call Level Interface (SAG CLI)

SAG CLI is a programming interface standard that defines the functions that are used to submit dynamic SQL clauses to a database server for execution. The ODBC interface is also based on SAG CLI. The Solid SQL API conforms to the SAG CLI standard.

Strict Durability

A transaction is fully (or "strictly") durable only if it becomes durable before returning from `Commit`. In other words, for durability to be strict, the user must not be told that the data has been committed unless

and until the transaction has been made durable (e.g. by writing it to a log file). In this situation, committed data is not lost if the server shuts down abnormally (e.g. due to a power failure).

Contrast this with "relaxed durability", which allows the user to be told that the data has been committed before the data has actually been made durable (e.g. written to a log file).

See also "Durability", "Relaxed Durability".

Stored Procedures

Database procedures allow programmers to split the application logic between the client and the server. These procedures are stored in the database, and they accept parameters in the activation call from the client application. This arrangement is used by intelligent transactions that are implemented with calls to stored procedures.

Structured Query Language (SQL)

SQL is a standardized query language designed for handling database requests and administration. The SQL syntax used in solidDB is based on the ANSI X3H2-1989 Level 2 standard including important ANSI X3H2-1992 (SQL-92) extensions. Refer to Appendix B, "solid sql syntax" in solidDB SQL Guide for a more formal definition of the syntax.

Subscription

See *solidDB SmartFlow Data Replication Guide*.

T

Table

A database table is a set of data elements, or fields, that is organized, defined and stored using a model of horizontal rows and vertical columns. The columns are identified by name, and the rows can be identified in various ways, often by the value appearing in a particular column which has been identified as the primary key.

A table has a specified number of columns but can have any number of rows. Besides the actual data rows, tables generally have associated with them some "header" information, such as constraints on the table or on the values within particular columns.

Transaction

Transaction is a group of SQL database commands regarded and executed as a single atomic entity. Ideally, a database system guarantees all of the ACID properties for each transaction. However, these properties are often relaxed to provide better performance.

Transaction Log File (Transaction Log)

This file holds a log of all committed operations executed by the database server. If a system crash occurs, the database server uses this log to recover all data inserted or modified after the latest checkpoint.

Transaction Isolation Level

When multiple users are using a database at the same time, one user's changes should only be visible to other users in controlled ways. For example, you might choose the "COMMITTED READ" isolation level, which means that you do not want to see any other user's changes (e.g. new records) that have not yet been committed yet. Or you might choose an isolation level that guarantees that if you look at the same table repeatedly in the same transaction, then you will see the same records each time. The ANSI standard for SQL defines 4 different levels of isolation. These are discussed in *solidDB Administration Guide* and are defined in the ANSI standard for SQL.

Note that solidDB supports both "transaction-level" isolation commands and "session-level" isolation commands. We refer to both as "transaction isolation commands".

For more information about transaction isolation levels, see the description of the SET TRANSACTION ISOLATION command (part of solidDB SQL Guide, Appendix B, Solid SQL Syntax), and chapter TRANSACTION ISOLATION Levels in solidDB SQL Guide.

Triggers

Triggers are pieces of logic that a solidDB server automatically executes when a user attempts to change the data in a table. When a user modifies data within the table, the trigger that corresponds to the command (such as insert, delete, or update) is activated.

Two-Tier Client/Server Architecture Model

Generally, the two-tier architecture refers to a client/server system, where a client application containing all the business logic is running on a workstation and a database server is taking care of data management.

Index

Symbols

"at" commands, 51
-x autoconvert, 216
-x convert, 216
1SafeMaxDelay (parameter), 177
2SafeAckPolicy (parameter), 177
@
 AT (@) sign, 80

A

abnormal shutdown
 recovering from, 48
AbortTimeOut (parameter), 199
Access Mode, 162
Access mode
 read-only, 162
 RO, 162
 RW, 162
 RW/Create, 162
 RW/Startup, 162
ACID, 319
AdaptiveRowsPerMessage (parameter), 199
ADMIN COMMAND
 abort, 303
 assertexit, 303
 backup, 303
 backuplist, 303
 backupserveron, 303
 checkpointing, 304
 close, 304
 commands, 301
 describe, 304
 errorcode, 304
 errorexit, 304
 filespec, 304
 help, 305
 hotstandby, 305
 info, 305
 makecp
 and checkpoint, 306
 memory, 307
 messages, 307
 monitor, 307
 netbackup, 307
 netbackuplist, 308
 notify, 308
 open, 308
 opt accelerator, 308
 parameter, 308
 perfmon, 309
 pid, 311
 pingtest, 311
 proctrace, 311
 protocols, 312
 runmerge, 312
 save parameters, 312
 setreadonlyflag, 312
 shutdown, 312
 sqllist, 312
 startmerge, 313
 status, 313
 thrinfor, 313
 throwout, 313
 trace, 313
 userid, 314
 userlist, 314
 usertrace, 316
 version, 317
ADMIN COMMAND 'perfmon'
 server performance, 31
ADMIN COMMAND 'report_report_filename'
 producing a report for troubleshooting, 37
ADMIN COMMAND 'status backup'
 querying last backup status, 30
ADMIN COMMAND 'status'
 querying database status, 27
ADMIN COMMAND 'throwout', 38
 disconnecting users, 30
ADMIN COMMAND 'userlist'

- querying for connected users, 29
- ADMIN COMMAND Syntax, 301
- admin commands
 - perfmon, 31
- AllowConnect (parameter), 199
- AllowDuplicateIndex (parameter), 195
- amount of memory used by in-memory tables and indexes, 305
- ANSI (reserved word), 88
- architecture
 - multithread processing, 14
- autocommit, 123
- autoconvert
 - command line option, 216
- automating administrative tasks, 18, 51
- AutoPrimaryAlone (parameter), 178

B

- B-tree, 11
- backup
 - and timed commands, 51
 - automating, 51
 - configuring and automating, 42
 - failed, 46
 - local, 39
 - monitoring and controlling, 45
 - network backup, 40
 - network backup server administration, 45
 - querying, 30
 - restoring, 47
 - typical problems, 46
 - What Happens During Backup?, 43
 - with timed command ("at" command), 52
- BackupBlockSize (parameter), 167
- BackupCopyIniFile (parameter), 168
- BackupCopyLog (parameter), 168
- BackupCopySolmsgOut (parameter), 168
- BackupDeleteLog (parameter), 168
- BackupDirectory (parameter), 168
- BackupDirectory (parameters), 66
- backups

- making manually, 38
- BackupStepsToSkip (parameter), 168
- bcktime, 305
- BLANKS
 - Solid SpeedLoader, 91
- BLOB, 11
- BLOBs
 - defining, 319
- BLOBs (Binary Large Objects), 23
 - defining, 22
- Blocksize (parameter), 22, 74
- BlockSize (parameter), 182, 185, 194
- Bonsai tree, 11
- Bonsai Tree, 121

C

- cache
 - database, 117
 - tuning cache, 117
- Cache, 320
- CacheSize (parameter), 183
- CacheSize (parameters), 65
- CAST, 297
- catalog
 - name criteria, 20
- Catalog, 320
 - defined, 320
- CatchupSpeedRate (parameter), 178
- CHARACTERSET
 - Solid SpeedLoader, 91
- CharPadding (parameter), 195
- checkpoint
 - and 'makecp' command, 306
- Checkpoint
 - defined, 321
- CheckpointDeleteLog (parameter), 168
- CheckpointInterval (parameter), 122, 169
- Checkpoints, 49
 - automatic daemon, 49
 - erasing automatically, 49
 - tuning, 122

- checkpoints
 - and timed commads, 51
 - automating, 51
 - forcing, 122
 - frequency, 122
- client-side configuration parameters, 211
- ClientReadTimeout (parameter), 212
- CLOB, 321
- close, 38
 - with timed command ("at" command), 52
- closing solidDB, 37
- clustering
 - data clustering, 11
- columns
 - setting LONG VARCHAR, 22
- Command Line Options, 215
- COMMIT statement
 - providing in application code, 125
 - troubleshooting, 125
- communication
 - between client and server, 129
 - selecting a protocol, 135
 - tracing problems, 145
- Communication protocol
 - defined, 321
- communication protocols, 135
 - DECnet, 141
 - Named Pipes, 139
 - NetBIOS, 140
 - selecting, 135
 - shared memory, 136
 - summary, 141
 - supported protocols, 135
 - TCP/IP, 136
 - UNIX Pipes, 138
- Communication Session Layer
 - described, 14
- communication tracing, 69
- configuration file, 155, 211
 - described, 21
 - on the client, 59
 - on the server, 59
- configuring
 - client-side configuration file, 59
 - configuration file, 59
 - default settings, 59
 - factory values, 59
 - managing parameters, 70
 - setting parameters, 70, 73
 - viewing parameter descriptions, 71
 - viewing parameters, 70
 - parameter settings, 59
 - server-side configuration file, 59
 - solid.ini, 59
 - example, 60
- Connect (parameter), 61, 178, 213
- connect string, 61
- connect strings for clients, 133
- Connecting to solidDB, 23
- ConnectionCheckInterval (parameter), 200
- connections
 - and committed transactions, 124
 - determining existing, 124
- ConnectStrForMaster (parameter), 208, 244
- ConnectTimeout (parameter), 179, 213
- ConnectTimeOut (parameter), 200
- control file
 - Solid SpeedLoader, 85, 88
- convert
 - command line option, 216
- converting database format, 216
- ConvertOrsToUnionsCount (parameter), 195
- CopyDirectory (parameter), 179
- cptime, 305
- Creating Checkpoints, 49
- Cursor
 - defined, 322
- CursorCloseAtTransEnd (parameter), 195

D

- D-table, 12
- Data definition language
 - defined, 322

Data manipulation language
 defined, 322

data sources
 defining in `sollid.ini`, 134

Data Sources, 142
 defining in `solid.ini`, 142

database, 63
 (see also index file)

- backing up, 38
- block size, 21
- cache, 117
- checking last backup status, 30
- checking overall status, 27
- closing, 50
 - automating, 51
- compacting, 52
- converting format, 216
- creating, 19
- currently connected users, 29
- decreasing database file size, 65
- defining objects, 22
- disconnecting a user, 30
- free space in, 305
- In-memory
 - configuring, 70
- location, 21, 22, 63
- login, 21, 23
 - (see also Connecting to `solidDB`)
- maximum size, 21
- monitoring, 31
- opening
 - automating, 51
- performance, 31
- querying last backup, 30
- recovery, 48
- restoring master and replica, 38
- several databases on one computer, 50
- shutting down, 37
- size, 20, 63
- troubleshooting, 31
- tuning cache, 117
- using in-memory database, 119

Database Management System
 defined, 322

Database maximum size, 21
 defining, 22

DatabaseSizeReportInterval (parameter), 200

DataDictionaryErrorMaxWait (parameter), 169

DATE
 Solid SpeedLoader, 92

dbfreesize, 305

DBMS
 defined, 322

dbsize, 305

DDL
 defined, 322

DECnet, 141

decreasing database file size, 65

DefaultStoreIsMemory (parameter), 170

Designing Intelligent transaction, 324

DigitTemplateChar (parameter), 185

DisableIdleMerge (parameter), 170

DisableOutput (parameter), 25, 200

Disabling message log output, 25

disconnecting users, 38

DML
 defined, 322

durability
 relaxed, 112, 327
 strict, 112, 329

Durability
 defined, 322

DurabilityLevel (parameter), 185

E

Echo (parameter), 201

EmulateOldTimestampDiff (parameter), 196

EnableHints (parameter), 196

ENCLOSURE
 Solid SpeedLoader, 93

entering timed commands, 51

environment variables
 SOLTRACE, 146

- SOLTRACEFILE, 146
- error codes
 - error handling, 221
- error handling
 - error codes, 221
 - Solid Communication Errors, 290
 - Solid Communication Warnings, 294
 - Solid Database Errors, 255
 - Solid Executable Errors, 266
 - Solid Procedure Errors, 294
 - Solid Server Errors, 287
 - Solid Sorter Errors, 298
 - Solid SpeedLoader Errors, 298
 - Solid SQL API Errors, 253
 - Solid SQL errors, 245
 - Solid Synchronization errors, 222
 - Solid System Errors, 267
 - Solid Table Errors, 270
- Error message codes, 26
- event
 - soldd and listing event definitions, 107
- Event alerts
 - defined, 323
- ExecRowsPerMessage (parameter), 201, 214
- executing system commands
 - automating, 51
- Execution Graph
 - described, 13
- ExtendIncrement (parameter), 120, 183
- external sorting, 118
 - specify a directory for External Sorting algorithm, 68

F

- FileFlush (parameter), 186
- FileNameTemplate (parameter), 187
- FileNameTemplate (parameters), 67
- FileSpec (parameter), 22, 63
- FileWriteFlushMode (parameter), 170
- ForceThreadsToSystemScope (parameter), 201

- Format of Configuration Parameter Names and Values, 160
- free space in database, 305

H

- HSBEnabled (parameter), 180

I

- I/O

- distributing, 120
 - tuning, 120

- IBMPC (reserved word), 88

- ImdbMemoryLimit (parameter), 189

- ImdbMemoryLowPercentage (parameter), 191

- imdbsize, 305

- ImplicitStart (parameter), 163

- import file

- Solid SpeedLoader, 85

- Incremental publication

- defined, 323

- Index

- defined, 323

- index file

- splitting to multiple disks, 63

- Info (parameter), 196

- Info (parameters), 69

- InfoFileFlush (parameter), 196

- InfoFileName (parameter), 196

- InfoFileSize (parameter), 196

- ini file

- Solid SpeedLoader, 86

- installing solidDB, 17

- intelligent join constraint transfer, 13

- Intelligent join constraint transfer

- defined, 323

- Intelligent transaction

- defined, 324

- design principles, 324

- example of, 324

- INTO_TABLE_PART

- Solid SpeedLoader, 93

IOTthreads (parameter), 170

isolation levels

- read committed, 114

- repeatable read, 114

- serializable, 115

IsolationLevel (parameter), 197

J

JDBC, 7, 8

L

Light Client, 61

Listen (parameter), 62, 164

listen name, 129, 131, 132, 133

Local backup, 39

LocalStartTasks (parameter), 202

Lock

- defined, 325

LockEscalationEnabled (parameter), 191

LockEscalationLimit (parameter), 192

LockHashSize (parameter), 170, 192

LockWaitTimeOut (parameter), 171

log file, 48

- Solid SpeedLoader, 86

log files

- solerror.out, 25

- solmsg.out, 25

LogDir (parameter), 188

LogEnabled (parameter), 188

logging

- transactions, 48

Logging and Transaction Durability, 111

Logical Data Source Names, 142

login, 21

- incorrect username or password, 21

logsize, 305

- from 'info' command, 305

LogWriteMode (parameter), 188

LongSequential SearchLimit (parameter), 172

M

M-table, 12

makecp, 122

- with timed command ("at" command), 52

manual administration, 18

master database

- backing up, 38

- restoring, 38

MasterStatementCache (parameter), 208

MaxBgTaskInterval (parameter), 202

MaxBlobExpressionSize (parameter), 197

- defining objects, 22

MaxCacheUsage (parameter), 193

MaxCacheUsePercent (parameter), 194

MaxConstraintLength (parameter), 203

MaxFilesTotal (parameter), 194

MaxLogSize (parameter), 180

MaxMemLogSize (parameter), 180

MaxMemPerSort (parameter), 194

MaxMergeParts (parameter), 172

MaxMergeTasks (parameter), 172

MaxNestedProcedures (parameter), 198

MaxNestedTriggers (parameter), 198

MaxOpenCursors (parameter), 204

MaxOpenFiles (parameter), 172

MaxPhysMsgLen (parameter), 164

MaxRPCDataLen (parameter), 204

MaxStartStatements (parameter), 204

memory

- physical, 117

- virtual, 117

memory allocation

- tuning, 116

MemoryReportDelta (parameter), 204

MemoryReportLimit (parameter), 204

MemorySizeReportInterval (parameter), 204

MergeInterval (parameter), 121, 172

MessageLogSize (parameter), 205

MinCheckpointTime (parameter), 122, 172

MinMergeTime (parameter), 172

MinSplitSize (parameter), 188

- monitoring
 - SolidConsole, 27
- monitorstate, 306
- MSWINDOWS (reserved word), 88
- Multithread Processing
 - described, 14
- multiversioning
 - Solid Bonsai Tree, 11

N

- Name (parameter), 205
- Named Pipes, 139
- NetBackup, 40
- NetBackupConnect (parameter), 173
- NetBackupConnectTimeout (parameter), 173
- NetBackupCopy Log (parameter), 173
- NetBackupCopy SolmsgOut (parameter), 173
- NetBackupCopyIniFile (parameter), 173
- NetBackupDeleteLog (parameter), 173
- NetBackupDirectory (parameter), 173
- NetBackupDirectory (parameters), 66
- NetBackupReadTimeout (parameter), 173
- NetBackupRootDir (parameter), 205
- netbackuptime, 305
- NetBIOS, 140
- NetcopyRpcTimeout (parameter), 180
- Network backup, 40
- Network Backup Directory
 - specifying, 66
- Network communication
 - communication session layer, 14
 - specifying tracing for, 69
 - troubleshooting, 152
- Network communications
 - Solid Network Services, 14
- network messages
 - tuning, 120
- Network name
 - defined, 326
- network names, 129, 131, 132, 133
 - adding, 131
 - clients, 133
 - DECnet, 141
 - defining, 60, 62
 - modifying, 131
 - Named Pipes, 139
 - NetBIOS, 140
 - removing, 132
 - shared memory, 136
 - TCP/IP, 136
 - UNIX Pipes, 138
 - viewing, 130, 131
- network trace facility, 145
- nmp, 139
- nmpipe, 139
- NoAssertMessages (parameter), 214
- node, 320
- non-graphical user interfaces
 - creating new database, 19
- NULLIF
 - Solid SpeedLoader, 91, 99
- NULLSTR
 - Solid SpeedLoader, 91
- numusers, 305

O

- ODBC, 7, 61
- open, 38
 - with timed command ("at" command), 52
- openstate, 306
- operating system
 - tuning, 117
- optimization
 - optimized sorts, 119
- optimized sorts, 119
- Optimizer Hints, 8

P

- parameters, 155, 211
 - BackupDirectory, 66
 - Blocksize, 22, 74
 - CacheSize, 65

- CheckpointInterval, 122
- Connect, 61
- ExtendIncrement, 120
- FileNameTemplate, 67
- FileSpec, 22, 63
- Info, 69
- Listen, 62
- MaxBlobExpressionSize, 22
- MergeInterval
 - setting, 121
- MinCheckpointTime, 122
- NetBackupDirectory, 66
- SortArraySize, 118
- Threads, 69
- TmpDir, 68
- Trace, 62, 69
- TraceFile, 62, 69
- passwords
 - criteria, 20
 - maximum number of characters, 20
- PCOEM (reserved word), 88
- perfmon (admin command), 31
- performance
 - diagnosing problems, 125
 - snapshot of, 31
 - tuning, 125
- performance monitoring
 - perfmon (admin command), 31
- performance tuning, 111
- Pessimistic (parameter), 173
- PessimisticTableUseNFetch (parameter), 205
- phantom, 115
- phantom updates
 - repeatable read, 115
 - serializable, 115
- Phantom updates
 - defined, 326
- physical memory, 117
- Ping facility, 148
- PingInterval (parameter), 181
- PingTimeout (parameter), 181
- POSITION

- Solid SpeedLoader, 99
- PreFlushPercent (parameter), 184
- PRESERVE BLANKS
 - Solid SpeedLoader, 93
- PrimaryAlone (parameter), 181
- primarystarttime, 306
- PrintMsgCode (parameter), 205
- problem reporting, 150
- ProcedureCache (parameter), 198
- Proprietary Interfaces, 8

Q

- Query
 - defined, 327
- Query Processing
 - described, 13
- querying database
 - ADMIN COMMAND 'status', 27

R

- RConnectLifetime (parameter), 164
- RConnectPoolSize (parameter), 164
- RConnectRPCTimeout (parameter), 165
- READ COMMITTED, 208
- ReadAhead (parameter), 184
- ReadBufSize (parameter), 165
- ReadLevelMaxTime (parameter), 174
- ReadMostlyLoadPercentAtPrimary (parameter), 164
- Readonly (parameter), 174
- ReadThreadMode (parameter), 205
- Record
 - defined, 327
- recovery, 112
 - automatic roll-forward, 38
- RefreshIsolationLevel (parameter), 208
- RefreshReadLevelRows (parameter), 209
- relaxed durability, 112
- Relaxed durability
 - defined, 327
- RelaxedMaxDelay (parameter), 189
- ReleaseMemoryAtShutdown (parameter), 193

RemoteStartTasks (parameter), 206
 REPEATABLE READ, 209
 replica database
 backing up, 38
 restoring, 38
 ReplicaRefreshLoad (parameter), 209
 reports
 automating, 51
 creating a continuous performance monitoring report, 32
 creating a report for troubleshooting, 37
 creating a status report, 37
 full list of perfmon counters, 32
 with timed command ("at" command), 52
 Restoring backups, 47
 RO
 access mode, 162
 roles for Database Administration
 for database administration, 17
 roll-forward recovery, 38
 RowsPerMessage (parameter), 206, 214
 RPC, 14
 RpcEventThresholdByteCount (parameter), 208
 running several servers, 50
 RW
 access mode, 162
 RW/Create
 access mode, 162
 RW/Startup
 access mode, 162

S
 SCAND7BIT (reserved word), 88
 Schema
 defined, 328
 scripts
 calling, 80
 executing SQL script from file, 81
 SearchBufferLimit (parameter), 174
 secondarystarttime, 306
 Sequence objects
 defined, 329
 sernum, 305
 Server name
 defined, 329
 server names, 129
 (see also network names)
 server-side configuration parameters, 155
 shared memory, 136
 shutdown, 38
 with timed command ("at" command), 52
 Shutting Down solidDB, 37
 Silent (parameter), 206
 SimpleOptimizerRulest (parameter), 198
 SocketLinger (parameter), 165
 SocketLingerTime (parameter), 165
 soldd, 10
 SOLDD, 106
 solerror.out
 described, 25
 solexp, 10
 SOLEXP, 104
 Solid Bonsai Tree, 123
 (see also Bonsai Tree)
 concurrency, 11
 index compression, 12
 multiversion, 11
 reducing size, 123
 Solid Communication Errors, 290
 Solid Communication Warnings, 294
 Solid Data Dictionary, 10, 106
 starting, 106
 Solid Data Disctionary
 defined, 10
 Solid Data Management Tools, 75
 solcon, 75
 soldd, 75
 solexp, 75
 SolidConsole, 75
 solload, 75
 Solid Database Errors, 255
 Solid executable
 -x execute command line option, 110

Solid Executable Errors, 266

Solid Export, 10, 104

- defined, 10
- starting, 104

Solid JDBC Driver

- troubleshooting, 152

Solid Light Client, 61, 133

Solid Network Services

- described, 14

Solid ODBC Driver

- troubleshooting, 151

Solid Procedure Errors, 294

Solid Remote Control (solcon), 76

Solid Remote Control (teletype)

- commands, 77
- starting, 76

Solid Server Errors, 287

Solid Sorter Errors, 298

Solid SpeedLoader

- control file, 85
- control file syntax, 88
- defined, 10
- described, 84
- import file, 85
- ini file, 86
- log file, 86

Solid SpeedLoader Errors, 298

Solid SQL API

- troubleshooting, 151

Solid SQL API Errors, 253

Solid SQL Editor (teletype), 78

- executing SQL statements, 80
- starting, 78

Solid SQL errors, 245

Solid SQL Optimizer

- described, 12

Solid Synchronization errors, 222

Solid System Errors, 267

Solid Table Errors, 270

Solid UNIFACE Driver

- troubleshooting, 152

solid.ini file

- configuration parameters, 155, 211
- configuring solidDB, 59
- described, 21

SolidConsole, 10

- administering multiple servers manually, 18
- described, 81
- interface features, 83
- monitoring, 27
- performing batch mode operations, 18
- starting, 82

solidDB

- Administering solidDB, 17
- command line options, 215
- components, 7
- Connecting to, 23
- executable program, 19
- installing, 17
- processes, 7
- programming interfaces, 7
- starting, 19

solidDB executable

- command line options, 215

SOLLOAD, 87

solmsg.out, 23

- described, 25

SOLTRACE

- environment variable, 146

SOLTRACEFILE

- environment variable, 146

SortArraySize (parameter), 118, 198

SorterEnabled (parameter), 194

sorting, 118

- optimized sorts, 119

space, 305

Special Roles for Database Administration, 17

SQL, 7

- defined, 330

SQL trace level

- setting, 69

SQL-89, 8

SQL-92, 8

SQL-99, 8

SQLInfo (parameter), 196
starting Solid Remote Control (teletype), 76
starting Solid SQL Editor (teletype), 78
Starting solidDB, 19
StartupForceMerge (parameter), 175
StatementMemoryTraceLimit (parameter), 206
storage tree
 described, 11
Store mode, 162
Stored procedures
 defined, 330
strict durability, 112
Strict durability
 defined, 329
Structured Query Language
 defined, 330
supported protocols, 130
SynchronizedWrite (parameter), 184
SyncWrite (parameter), 189
syntax
 ADMIN COMMAND, 301
Syntax Analysis
 described, 13
SYS_ADMIN_ROLE
 for database administration, 17
SYS_CONSOLE_ROLE
 for database administration, 18
SYS_R_MAXBYTES_IN (parameter)
 described, 236
SYS_R_MAXBYTES_OUT (parameter)
 message length, 236
SYS_SYNC_ADMIN_ROLE
 for database administration, 18
SYS_SYNC_REGISTER_ROLE
 for database administration, 18
system
 with timed command ("at" command), 52

T

Table
 defined, 330

TableLockWaitTimeout (parameter), 175
TCP/IP, 8, 136
TcpKeepAlive (parameter), 165
TcpKeepAliveIdleTime (parameter), 165
TcpKeepAliveProbeCount (parameter), 166
TcpKeepAliveProbeInterval (parameter), 166
TERMINATION
 Solid SpeedLoader, 97
thread, 14
 dedicated, 15
 general purpose, 15
 setting for processing, 68
 types of, 15
Threads (parameter), 206
Threads (parameters), 69
throwing out users
 automating, 51
throwout, 30
 with timed command ("at" command), 52
throwout all, 38
TIME
 Solid SpeedLoader, 92
timed commads
 and backups, 51
 and checkpoints, 51
timed commands, 51
 at, 51
TIMESTAMP
 Solid SpeedLoader, 92
TimestampDisplaySize19 (parameter), 198
TmpDir (parameters), 68
TmpDir_[1... N] (parameter), 194
TmpDir_[1...N], 68
Trace (parameter), 62, 167, 213
Trace (parameters), 69
trace files
 described, 25
Trace files, 26
TraceFile (parameter), 62, 167, 213
TraceFile (parameters), 69
TraceLogSize (parameter), 207
TraceSecDecimals (parameter), 207

- tracestate, 305
- tracing communication, 145
- Tracing Failed Login Attempts, 26
- transaction
 - logging, 48
- Transaction
 - defined, 330
- Transaction isolation level
 - defined, 331
- Transaction log
 - defined, 331
- transaction log files
 - specifying directory, 67
- Transaction Logging, 48
 - Overwriting, 48
 - Ping-pong, 48
- TransactionEarlyValidate (parameter), 175
- TransactionHashSize (parameter), 175
- transactions
 - committing to reduce Bonsai Tree size, 123
- tries, 12
- TriggerCache (parameter), 198
- tuning
 - I/O, 120
 - memory allocation, 116
 - network messages, 120
 - operating system, 117
- Tuning
 - checkpoints, 122
- tuning cache, 117
- Tuning checkpoints, 122

U

- UNIX Pipes, 138
- uptime, 305
- usernames
 - criteria, 19
 - default, 20
 - maximum number of characters, 20
- users
 - throwing out, 51

V

- VersionedPessimisticReadCommitted (parameter), 176
- VersionedPessimisticRepeatableRead (parameter), 176
- viewing message log, 25
- Viewing the solidDB Message Log, 25
- virtual memory, 117

W

- Windows Registry
 - data sources, 143
- WriteBufSize (parameter), 167
- WriterIOThreads (parameter), 176