



What's New in SQL Anywhere® 10

Last modified: April 2006

Copyright © 2006 iAnywhere Solutions, Inc. Portions copyright © 2006 Sybase, Inc. All rights reserved.

iAnywhere Solutions, Inc. is a subsidiary of Sybase, Inc.

iAnywhere grants you permission to use this document for your own informational, educational, and other non-commercial purposes; provided that (1) you include this and all other copyright and proprietary notices in the document in all copies; (2) you do not attempt to “pass-off” the document as your own; and (3) you do not modify the document. You may not publish or distribute the document or any portion thereof without the express prior written consent of iAnywhere.

This document is not a commitment on the part of iAnywhere to do or refrain from any activity, and iAnywhere may change the content of this document at its sole discretion without notice. Except as otherwise provided in a written agreement between you and iAnywhere, this document is provided “as is”, and iAnywhere assumes no liability for its use or any inaccuracies it may contain.

iAnywhere®, Sybase®, and the marks listed at <http://www.ianywhere.com/trademarks> are trademarks of Sybase, Inc. or its subsidiaries. ® indicates registration in the United States of America.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Contents

About This Manual	vii
SQL Anywhere documentation	viii
Documentation conventions	xi
Finding out more and providing feedback	xiv
1 What's New in Version 10.0.0	1
SQL Anywhere	3
MobiLink	68
QAnywhere	91
SQL Remote	96
UltraLite	97
Sybase Central and Interactive SQL	113
Documentation enhancements	119
Product-wide features	120
Beta notes	123
2 What's New in Version 9.0.2	125
New features in version 9.0.2	126
Behavior changes in version 9.0.2	146
3 What's New in Version 9.0.1	153
New features in version 9.0.1	154
Behavior changes in version 9.0.1	172
4 What's New in Version 9.0.0	177
New features in version 9.0.0	178
Behavior changes in version 9.0	201
5 What's New in Version 8.0.2	211
New features in version 8.0.2	212
Behavior changes in version 8.0.2	223
6 What's New in Version 8.0.1	227
New features in version 8.0.1	228
Behavior changes in version 8.0.1	235
7 What's New in Version 8.0.0	239
New features in version 8	240
Behavior changes in version 8	266

8	What's New in Version 7.0.3	277
	New features	278
	Behavior changes	279
9	What's New in Version 7.0.2	281
	New features in version 7.0.2	282
	Behavior changes in version 7.0.2	286
10	What's New in Version 7.0.1	287
	New features in version 7.0.1	288
11	What's New in Version 7.0.0	295
	New features in version 7.0.0	296
	Behavior changes in version 7.0.0	307
12	What's New in Version 6.0.3	311
	New features in version 6.0.3	312
	Behavior changes in version 6.0.3	319
13	What's New in Version 6.0.2	321
	New features in version 6.0.2	322
	Behavior changes in version 6.0.2	325
14	What's New in Version 6.0.1	327
	New features in version 6.0.1	328
	New features in SQL Remote	331
	Behavior changes	332
15	Upgrading to SQL Anywhere 10	333
	Upgrading SQL Anywhere	334
	Upgrading MobiLink	342
	Upgrading QAnywhere	348
	Upgrading UltraLite	349
	Upgrading SQL Remote	359
16	Upgrading Version 5 Applications	361
	Understanding version 5 upgrades	362
	Review of SQL Anywhere Version 5 architecture	365
	Upgrading embedded SQL applications	368
	Upgrading ODBC applications	373
	Using the compatibility library	378
	Capturing dbclient command information	384
	Upgrading databases	386
	Upgrading version 5 SQL Remote installations	387

About This Manual

Subject	This book describes new features in SQL Anywhere 10 and in previous versions of the software.
Audience	This manual is for users of previous versions who want to find out what is new and different in this version of the software.

SQL Anywhere documentation

The SQL Anywhere documentation

This book is part of the SQL Anywhere documentation set. This section describes the books in the documentation set and how you can use them.

The complete SQL Anywhere documentation is available in two forms: an online form and as separate Adobe Portable Document Format (pdf) files for each book. Both forms of the documentation contain the identical information and consist of the following books:

- ◆ **Introducing SQL Anywhere** This book introduces SQL Anywhere—the comprehensive package that provides data management and enterprise data synchronization, enabling the rapid development of database-powered applications for server, desktop, mobile, and remote office environments.
- ◆ **What's New in SQL Anywhere 10** This book describes new features in SQL Anywhere 10 and in previous versions of the software.
- ◆ **SQL Anywhere Database Administration Guide** This book covers material related to running, managing, and configuring databases. It describes database connections, the database server, database files, security, backup procedures, security, and replication with Replication Server, as well as administration utilities and options.
- ◆ **SQL Anywhere SQL User's Guide** This book describes how to design and create databases; how to import, export, and modify data; how to retrieve data; and how to build stored procedures and triggers.
- ◆ **SQL Anywhere Reference** This book provides a complete reference for the SQL language used by SQL Anywhere. It also describes the SQL Anywhere system views and procedures.
- ◆ **SQL Anywhere Programming Guide** This book describes how to build and deploy database applications using the C, C++, and Java programming languages, as well as Visual Studio .NET. Users of tools such as Visual Basic and PowerBuilder can use the programming interfaces provided by those tools.
- ◆ **SQL Anywhere Error Messages** This book provides a complete listing of Adaptive Server Anywhere error messages together with diagnostic information.
- ◆ **Getting Started with MobiLink** This manual introduces MobiLink, a session-based relational-database synchronization system. MobiLink technology allows two-way replication and is well suited to mobile computing environments.

-
- ◆ **MobiLink Administration Guide** This manual describes how to set up and administer MobiLink applications.
 - ◆ **MobiLink Clients** This manual describes MobiLink, a session-based relational-database synchronization system. MobiLink technology allows two-way replication and is well suited to mobile computing environments.
 - ◆ **MobiLink Server-Initiated Synchronization User's Guide** This manual describes MobiLink server-initiated synchronization, a feature of MobiLink that allows you to initiate synchronization or other remote actions from the consolidated database.
 - ◆ **QAnywhere User's Guide** This manual describes QAnywhere, which defines a messaging platform for mobile and wireless clients as well as traditional desktop and laptop clients.
 - ◆ **SQL Remote User's Guide** This book describes the SQL Remote data replication system for mobile computing, which enables sharing of data between a SQL Anywhere consolidated database and many SQL Anywhere remote databases using an indirect link such as email or file transfer.
 - ◆ **SQL Anywhere 10 Help** This manual provides context-sensitive help for the Connect dialog, the Query Editor, the MobiLink Monitor, the SQL Anywhere Console utility, the Index Consultant. It also contains context-sensitive help for all the property sheets and dialog boxes that are available in Sybase Central, as well as for MobiLink wizards.
 - ◆ **UltraLite Database User's Guide** This manual introduces the UltraLite database system for small devices.
 - ◆ **UltraLite for Crossfire User's Guide** This manual describes UltraLite for Crossfire. With UltraLite for Crossfire you can develop and deploy database applications to handheld, mobile, or embedded devices, running Palm OS or Windows CE.
 - ◆ **UltraLite.NET User's Guide** This manual describes UltraLite.NET. With UltraLite.NET you can develop and deploy database applications to computers, or handheld, mobile, or embedded devices.
 - ◆ **UltraLite for M-Business Anywhere User's Guide** This manual describes UltraLite for M-Business Anywhere. With UltraLite for M-Business Anywhere you can develop and deploy web-based database applications to handheld, mobile, or embedded devices, running Palm OS, Windows CE, or Windows XP.

-
- ◆ **UltraLite C/C++ User's Guide** This manual describes UltraLite C and C++ programming interfaces. With UltraLite you can develop and deploy database applications to handheld, mobile, or embedded devices.

Documentation formats SQL Anywhere provides documentation in the following formats:

- ◆ **Online documentation** The online documentation contains the complete SQL Anywhere documentation, including the books and the context-sensitive help for SQL Anywhere tools. The online documentation is updated with each maintenance release of the product, and is the most complete and up-to-date source of documentation.

To access the online documentation on Windows operating systems, choose Start ► Programs ► SQL Anywhere 10 ► Online Books. You can navigate the online documentation using the HTML Help table of contents, index, and search facility in the left pane, as well as using the links and menus in the right pane.

To access the online documentation on Unix operating systems, see the HTML documentation under your SQL Anywhere installation or on your installation CD.

- ◆ **PDF books** The complete set of SQL Anywhere books is provided as a set of Adobe Portable Document Format (pdf) files, viewable with Adobe Reader.

On Windows, the PDF books are accessible from the online books via the PDF link at the top of each page, or from the Windows Start menu (Start ► Programs ► SQL Anywhere 10 ► Online Books - PDF Format).

On Unix, the PDF books are accessible on your installation CD.

Documentation conventions

This section lists the typographic and graphical conventions used in this documentation.

Syntax conventions

The following conventions are used in the SQL syntax descriptions:

- ◆ **Keywords** All SQL keywords appear in uppercase, like the words `ALTER TABLE` in the following example:

ALTER TABLE [*owner*.]*table-name*

- ◆ **Placeholders** Items that must be replaced with appropriate identifiers or expressions are shown like the words *owner* and *table-name* in the following example:

ALTER TABLE [*owner*.]*table-name*

- ◆ **Repeating items** Lists of repeating items are shown with an element of the list followed by an ellipsis (three dots), like *column-constraint* in the following example:

ADD *column-definition* [*column-constraint*, ...]

One or more list elements are allowed. In this example, if more than one is specified, they must be separated by commas.

- ◆ **Optional portions** Optional portions of a statement are enclosed by square brackets.

RELEASE SAVEPOINT [*savepoint-name*]

These square brackets indicate that the *savepoint-name* is optional. The square brackets should not be typed.

- ◆ **Options** When none or only one of a list of items can be chosen, vertical bars separate the items and the list is enclosed in square brackets.

[**ASC** | **DESC**]

For example, you can choose one of `ASC`, `DESC`, or neither. The square brackets should not be typed.

- ◆ **Alternatives** When precisely one of the options must be chosen, the alternatives are enclosed in curly braces and a bar is used to separate the options.

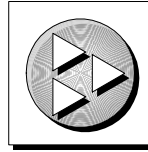
[**QUOTES** { **ON** | **OFF** }]

If the `QUOTES` option is used, one of `ON` or `OFF` must be provided. The brackets and braces should not be typed.

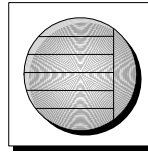
Graphic icons

The following icons are used in this documentation.

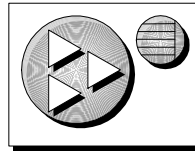
- ◆ A client application.



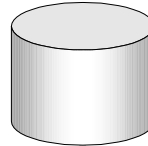
- ◆ A database server, such as SQL Anywhere.



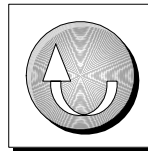
- ◆ An UltraLite application.



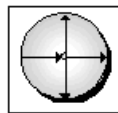
- ◆ A database. In some high-level diagrams, the icon may be used to represent both the database and the database server that manages it.



- ◆ Replication or synchronization middleware. These assist in sharing data among databases. Examples are the MobiLink Synchronization Server and the SQL Remote Message Agent.



- ◆ A Sybase Replication Server



-
- ♦ A programming interface.



Finding out more and providing feedback

Finding out more

Additional information and resources, including a code exchange, are available at the iAnywhere Developer Network at <http://www.ianywhere.com/developer/>.

If you have questions or need help, you can post messages to the iAnywhere Solutions newsgroups listed below.

When you write to one of these newsgroups, always provide detailed information about your problem, including the build number of your version of SQL Anywhere. You can find this information by entering **dbeng10 -v** at a command prompt.

The newsgroups are located on the *forums.sybase.com* news server. The newsgroups include the following:

- ◆ [sybase.public.sqlanywhere.general](#)
- ◆ [sybase.public.sqlanywhere.linux](#)
- ◆ [sybase.public.sqlanywhere.mobilink](#)
- ◆ [sybase.public.sqlanywhere.product_futures_discussion](#)
- ◆ [sybase.public.sqlanywhere.replication](#)
- ◆ [sybase.public.sqlanywhere.ultralite](#)
- ◆ [ianywhere.public.sqlanywhere.qanywhere](#)

Newsgroup disclaimer

iAnywhere Solutions has no obligation to provide solutions, information, or ideas on its newsgroups, nor is iAnywhere Solutions obliged to provide anything other than a systems operator to monitor the service and ensure its operation and availability.

iAnywhere Solutions Technical Advisors as well as other staff assist on the newsgroup service when they have time available. They offer their help on a volunteer basis and may not be available on a regular basis to provide solutions and information. Their ability to help is based on their workload.

Feedback

We would like to receive your opinions, suggestions, and feedback on this documentation.

You can email comments and suggestions to the SQL Anywhere documentation team at iasdoc@ianywhere.com. Although we do not reply to emails sent to that address, we read all suggestions with interest.

In addition, you can provide feedback on the documentation and the software through the newsgroups listed above.

CHAPTER 1

What's New in Version 10.0.0

About this chapter

This chapter provides an overview of the new features and behavior changes introduced in SQL Anywhere 10. It also lists deprecated features, which are supported in the current version but will not be supported in the next major release of SQL Anywhere.

Note

In version 10.0.0, Adaptive Server Anywhere was renamed SQL Anywhere.

SQL Anywhere highlights

- ◆ [“New features” on page 3](#)
- ◆ [“Behavior changes” on page 36](#)
- ◆ [“Deprecated and discontinued features” on page 60](#)

MobiLink

- ◆ [“New features” on page 68](#)
- ◆ [“Behavior changes and deprecated features” on page 80](#)

QAnywhere

- ◆ [“New features” on page 91](#)
- ◆ [“Behavior changes and deprecated features” on page 95](#)

SQL Remote

- ◆ [“New features” on page 96](#)
- ◆ [“Behavior changes and deprecated features” on page 96](#)

UltraLite

- ◆ [“New features” on page 97](#)
- ◆ [“Behavior changes and deprecated features” on page 110](#)

Sybase Central and Interactive SQL

- ◆ [“New features” on page 113](#)
- ◆ [“Behavior changes and deprecated features” on page 116](#)

Documentation enhancements

- ◆ [“Documentation enhancements” on page 119](#)

Deprecated feature lists subject to change

As with all forward-looking statements, the lists of deprecated features are not guaranteed to be complete and are subject to change.

Contents

Topic:	page
SQL Anywhere	3
MobiLink	68
QAnywhere	91

Topic:	page
SQL Remote	96
UltraLite	97
Sybase Central and Interactive SQL	113
Documentation enhancements	119
Product-wide features	120
Beta notes	123

SQL Anywhere

New features

Following is a list of additions to SQL Anywhere databases and database servers introduced in version 10.0.0.

Main features

- ◆ **Support for intra-query parallelism to improve performance** The database server now supports the use of multiple processors for processing a single query. Intra-query parallelism is beneficial when the number of simultaneously executing queries is less than the number of available processors. For information on how to enable this feature, and control the database server's use of it, see [“max_query_tasks option \[database\]”](#) [*SQL Anywhere Database Administration Guide*, page 414].
- ◆ **Support for database mirroring** SQL Anywhere now supports database mirroring, which is a mechanism to increase the availability of a database. It involves using either two or three database servers running on separate computers and communicating with each other in either synchronous or asynchronous mode. See [“Understanding database mirroring”](#) [*SQL Anywhere Database Administration Guide*, page 792].

The following features have been added to support database mirroring:

- [“synchronize_mirror_on_commit option \[database\]”](#) [*SQL Anywhere Database Administration Guide*, page 443]
- Alternate server names for database servers. See [“-sn database option”](#) [*SQL Anywhere Database Administration Guide*, page 195] and [“START DATABASE statement”](#) [*SQL Anywhere Reference*, page 650].
- ServerName property
- AlternateServerName property
- ◆ **Support for snapshot isolation** When you use snapshot isolation, the database keeps a copy of the original data while a user is changing it, and makes the original data available to other users who want to read it. Snapshot isolation is completely transparent to users, and can help reduce deadlocks and lock contentions. See [“Snapshot isolation”](#) [*SQL Anywhere SQL User's Guide*, page 132].

The following features have been added or enhanced to support snapshot isolation:

- ◆ [“allow_snapshot_isolation option \[database\]”](#) [*SQL Anywhere Database Administration Guide*, page 371]

- ◆ “isolation_level option [compatibility]” [*SQL Anywhere Database Administration Guide*, page 401]
 - ◆ “sa_snapshots system procedure” [*SQL Anywhere Reference*, page 917]
 - ◆ “sa_transactions system procedure” [*SQL Anywhere Reference*, page 922]
 - ◆ LockCount, SnapshotCount, SnapshotIsolationState, and VersionStorePages database properties
 - ◆ LockCount and SnapshotCount connection properties
 - ◆ Version Store Pages Performance Monitor statistic
 - ◆ “SET statement [T-SQL]” [*SQL Anywhere Reference*, page 633]
 - ◆ “OPEN statement [ESQL] [SP]” [*SQL Anywhere Reference*, page 574]
 - ◆ “The ValuePtr parameter” [*SQL Anywhere SQL User’s Guide*, page 140]
- ◆ **Support for application profiling and diagnostic tracing** Existing application profiling capabilities, such as stored procedure profiling and request logging, have been integrated into a single, unified interactive interface the SQL Anywhere plug-in for Sybase Central. When you profile your application from Sybase Central, recommendations are provided to help you improve database performance.

Following is a list of procedures and function created to support this feature:

- ◆ “sa_save_trace_data system procedure” [*SQL Anywhere Reference*, page 906]
- ◆ “sa_set_tracing_level system procedure” [*SQL Anywhere Reference*, page 916]
- ◆ “sa_remove_tracing_data system procedure” [*SQL Anywhere Reference*, page 902]
- ◆ “TRACED_PLAN function [Miscellaneous]” [*SQL Anywhere Reference*, page 239]

Following is a list of statements created to support this feature:

- ◆ “ATTACH TRACING statement” [*SQL Anywhere Reference*, page 316]
- ◆ “DETACH TRACING statement” [*SQL Anywhere Reference*, page 473]
- ◆ “REFRESH TRACING statement” [*SQL Anywhere Reference*, page 598]

Two new database properties, SendingTracingTo and ReceivingTracingFrom have been added to support this feature. See “Database-level properties” [*SQL Anywhere Database Administration Guide*, page 492].

A new dbunload option, make auxiliary catalog (-k), has also been added. Specifying this option populates the sa_diagnostic_auxiliary_catalog table. This table maps database object IDs for tables, users, procedures, and so on, from the source database to the tracing database. See “The Unload utility” [*SQL Anywhere Database Administration Guide*, page 642].

- ◆ **Support for materialized views** To improve performance in environments where the database is large and frequent queries result in repetitive aggregation and join operations on large amounts of data, SQL Anywhere now supports the use of materialized views. See [“Working with materialized views” \[SQL Anywhere SQL User's Guide, page 71\]](#).

The database server has been enhanced to automatically decide, based on cost, which materialized views can be used to answer parts of a query instead of using base tables referenced directly by the query. See [“Improving performance with materialized views” \[SQL Anywhere SQL User's Guide, page 514\]](#).

Two new system tables, ISYSMVOPTION and ISYSMVOPTIONNAME, have been added to store information about materialized views. See [“SYSMVOPTION system view” \[SQL Anywhere Reference, page 761\]](#) and [“SYSMVOPTIONNAME system view” \[SQL Anywhere Reference, page 762\]](#).

- ◆ **Support for NCHAR data** SQL Anywhere now supports the NCHAR data type. NCHAR data types are used for storing Unicode character data. See [“NCHAR data type” \[SQL Anywhere Reference, page 52\]](#).

The following new functions have been added in support of NCHAR:

- ◆ [“UNISTR function \[String\]” \[SQL Anywhere Reference, page 242\]](#)
- ◆ [“CONNECTION_EXTENDED_PROPERTY function \[String\]” \[SQL Anywhere Reference, page 116\]](#)
- ◆ [“UNICODE function \[String\]” \[SQL Anywhere Reference, page 241\]](#)
- ◆ [“NCHAR function \[String\]” \[SQL Anywhere Reference, page 184\]](#)
- ◆ [“TO_CHAR function \[String\]” \[SQL Anywhere Reference, page 237\]](#)
- ◆ [“TO_NCHAR function \[String\]” \[SQL Anywhere Reference, page 237\]](#)

The following functions SORTKEY and COMPARE functions have new parameters to support the NCHAR data type:

- ◆ [“SORTKEY function \[String\]” \[SQL Anywhere Reference, page 223\]](#)
- ◆ [“COMPARE function \[String\]” \[SQL Anywhere Reference, page 113\]](#)

SQL Anywhere now properly sorts multi-byte character sets when using the Unicode Collation Algorithm (UCA).

The Initialization utility (dbinit) and Unload (dbunload) utilities also have new options to support the NCHAR data type. See [“The Initialization utility” \[SQL Anywhere Database Administration Guide, page 573\]](#), and [“The Unload utility” \[SQL Anywhere Database Administration Guide, page 642\]](#).

SQL Anywhere now uses International Components for Unicode (ICU) for Unicode support. See [“International Languages and Character Sets” \[SQL Anywhere Database Administration Guide, page 281\]](#).

To support ICU and the handling of NCHAR data, the following property changes have been made:

- A new NcharCharSet database and connection extended property has been added. This property returns the NCHAR character set in use by the database or connection.
- A new AccentSensitive database property has been added. This property returns the status of the accent sensitivity feature.
- The CharSet database and connection properties are now extended properties.

See “Database-level properties” [[SQL Anywhere Database Administration Guide, page 492](#)], and “Connection-level properties” [[SQL Anywhere Database Administration Guide, page 458](#)].

- ◆ **Improved control over how indexes are created** When an application creates a referential integrity constraint (primary key, foreign key, or unique constraint), the database server enforces the constraint by implicitly creating an index on the columns that make up the key of the constraint. The database server now allows you to specify how the index is created. You can specify the order of columns in the constraint key and the sequencing of values (ascending or descending) for each column in the index. In addition, there is no requirement that the order and sequencing of columns in a foreign key match the same for the corresponding primary key or unique constraint.

Additional improvements include:

- The primary key order can now be changed without having to reorder the columns in the table.
 - The sequencing of columns in all constraint indexes can be specified to match application requirements.
 - Foreign key indexes can now be tailored to match the application requirements for the foreign key table without being tied to the primary table design.
 - Foreign keys can now have unique constraints.
- ◆ **Internal performance enhancements** Following are performance enhancements additional to those inherent in the new features of this release:

- **Improved performance in the server**
[under construction] Expression building and manipulation has been optimized to significantly improve throughput.
- **Improved scalability of memory intensive operators**
[under construction]

- ◆ **Support for view dependencies** The catalog now stores information about the dependencies of views. Specifically, the catalog keeps track of the views, tables and columns upon which each view in the database depends. When you make an alteration to an object upon which a view depends, the database server automatically performs additional operations to ensure that the view definition is not left in a state where it could return incorrect results. See “[View dependencies](#)” [*SQL Anywhere SQL User's Guide*, page 67].

Two new tables, `ISYSDEPENDENCY` and `ISYSOBJECT`, have been added to store information about system objects and their dependencies. See “[SYSDEPENDENCY system view](#)” [*SQL Anywhere Reference*, page 747] and “[SYSOBJECT system view](#)” [*SQL Anywhere Reference*, page 762].

- ◆ **Support for lazy checkpoints** The database server can now initiate a checkpoint and perform other operations while it takes place. Previously, all activity would stop while the checkpoint took place. If a checkpoint is already in progress, then any operation like an `ALTER TABLE` or `CREATE INDEX` that wants to initiate a new checkpoint needs to wait for the last one to finish.
- ◆ **Locking enhancements** The following enhancements have been made to locking:
 - **Support for intent locks** Transactions can now acquire intent locks, which indicate an intention to modify a particular row. See “[Intent locks](#)” [*SQL Anywhere SQL User's Guide*, page 171].
 - **drop for support of WITH HASH SIZE** [under construction]
 - **new index implementation** [under construction]
 - **support for snapshot isolation** [under construction] (leave index entries in the index until committed/cleaned up)
 - **key range locks no longer required** [under construction]
at end of index for row insertions
- ◆ **Improved BLOB storage control and performance** You can now control the amount of a BLOB value that is stored in a table row (inline). You can also control whether to index BLOB values. These enhancements improve searching for, and accessing, BLOBs, and are made available through three new clauses in the `CREATE TABLE` and `ALTER TABLE` statements: `INLINE`, `PREFIX`, and `[NO] INDEX`. BLOB values can now be shared within, or among rows of the same table, reducing storage requirements by eliminating the need to store duplicate BLOB values. See “[BLOB storage](#)” [*SQL Anywhere SQL User's*

[Guide, page 25](#)], “[CREATE TABLE statement](#)” [[SQL Anywhere Reference, page 427](#)], and “[ALTER TABLE statement](#)” [[SQL Anywhere Reference, page 303](#)].

- ◆ **Support for column compression** You can now compress individual columns in a table. Compression is achieved using the deflate compression algorithm. This is the same compression used by the COMPRESS function, and is also the algorithm used in Windows .zip files. See “[CREATE TABLE statement](#)” [[SQL Anywhere Reference, page 427](#)] and “[ALTER TABLE statement](#)” [[SQL Anywhere Reference, page 303](#)].
- ◆ **Support for table encryption** Instead of encrypting an entire database to secure data, you can now encrypt individual tables in the database. Table encryption must be enabled in the database when it is initialized. See “[Table encryption](#)” [[SQL Anywhere Database Administration Guide, page 833](#)].

Administration

- ◆ **Database options** The following database options have been added or have enhanced functionality:
 - **ansi_blanks option** This option controls truncation warnings sent to the client if the expression being fetched is CHAR or NCHAR and it is being fetched into a CHAR or NCHAR host variable. See “[ansi_blanks option \[compatibility\]](#)” [[SQL Anywhere Database Administration Guide, page 372](#)].
 - **ansi_substring database option** This option has controls the behavior of the SUBSTRING function. By default, the behavior of the SUBSTRING function now corresponds to ANSI/ISO SQL/99 behavior. A negative or zero start offset is treated as if the string were padded on the left with non-characters, and gives an error if a negative length is provided. See “[ansi_substring option \[compatibility\]](#)” [[SQL Anywhere Database Administration Guide, page 374](#)].
 - **collect_statistics_on_dml_updates database option** Controls the gathering of statistics during the execution of DML statements (INSERT, DELETE, and UPDATE). See “[collect_statistics_on_dml_updates option \[database\]](#)” [[SQL Anywhere Database Administration Guide, page 383](#)].
 - **default_dbspace option** This option allows you to specify the dbspace where your tables are created. See “[default_dbspace option \[database\]](#)” [[SQL Anywhere Database Administration Guide, page 391](#)].
 - **max_temp_space database option** You can specify the maximum amount of temporary space a connection can use with the

max_temp_space option. See “[max_temp_space option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 417].

- **materialized_view_optimization database option** This option controls optimizer's use of materialized views. See “[materialized_view_optimization option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 411].
 - **oem_string database option** You can store information in the header page of a database file using the oem_string database option that can be accessed by applications. This string can be used for such purposes as storing version information, or validating that the database file is intended for your application. See “[oem_string option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 420].
 - **request_timeout database option** This option allows you to specify the maximum time a single request can run to help prevent connections from consuming a significant amount of server resources for a long period of time. See “[request_timeout option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 436].
 - **synchronize_mirror_on_commit option** This option controls when database changes are assured to have been sent to a mirror server when database running mirroring in asynchronous or asyncfullpage mode. See “[synchronize_mirror_on_commit option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 443].
 - **uuid_has_hyphens database option** This option has controls the formatting of uniqueidentifier values when they are converted to strings. See “[uuid_has_hyphens option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 452].
 - **verify_password_function database option** The verify_password_function database option allows you to specify a function that can be used to implement password rules. The function is called on a GRANT CONNECT statement. See “[verify_password_function option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 453].
- ◆ **New server options** The following new server options have been added:
- **-cm server option** This server option allows you to specify the amount of address space allocated for Address Windowing Extensions (AWE) on Windows 2000, Windows XP, and Windows Server 2003. See “[-cm server option](#)” [*SQL Anywhere Database Administration Guide*, page 135].
 - **-dh server option** This server option makes a database undetectable when the Server Enumeration utility (dblocate) is run against the

server. See “[-dh database option](#)” [*SQL Anywhere Database Administration Guide*, page 192].

- **-dt server option** This server option allows you to specify the directory where temporary files are stored. This option cannot be specified for database servers using shared memory connections on Unix. See “[-dt server option](#)” [*SQL Anywhere Database Administration Guide*, page 140].
- **-gtc server option** This option lets you control the number of threads that can run concurrently on a CPU. See “[-gtc server option](#)” [*SQL Anywhere Database Administration Guide*, page 154].
- **-ot server option** When you specify this server option, the log file is truncated before any messages are written to it. See “[-ot server option](#)” [*SQL Anywhere Database Administration Guide*, page 162].
- ◆ **New properties and Performance Monitor statistics** Several new connection, server, and database properties, as well as new Performance Monitor statistics, have been added to help you administer your database.
 - **Connection properties** The following connection properties have been added in this release:
 - ◆ ApproximateCPUTime
 - ◆ CacheFree
 - ◆ CacheFile
 - ◆ CacheFileDirty
 - ◆ CacheAllocated
 - ◆ CachePanics
 - ◆ CacheScavenges
 - ◆ CacheScavengesVisited
 - ◆ CarverHeapPages
 - ◆ CurrentProcedure
 - ◆ CurrentLineNumber
 - ◆ HeapsCarver
 - ◆ HeapsLocked
 - ◆ HeapsQuery
 - ◆ HeapsRelocatable
 - ◆ HttpServiceName
 - ◆ LastPlanText
 - ◆ LockCount
 - ◆ LockedCursorPages
 - ◆ max_temp_space
 - ◆ MultiPageAllocs
 - ◆ NcharCharSet
 - ◆ oem_string
 - ◆ QueryHeapPages

- ◆ ReqCountActive
- ◆ ReqCountBlockContention
- ◆ ReqCountBlockLock
- ◆ ReqCountBlockIO
- ◆ ReqCountUnscheduled
- ◆ ReqTimeActive
- ◆ ReqTimeBlockContention
- ◆ ReqTimeBlockIO
- ◆ ReqTimeBlockLock
- ◆ ReqTimeUnscheduled
- ◆ RequestStatus
- ◆ RequestsReceived
- ◆ RetryConnectionTimeout
- ◆ SnapshotCount

For more information about connection properties, see

[“Connection-level properties” \[SQL Anywhere Database Administration Guide, page 458\]](#).

- **Server properties** The following server properties have been added in this release:

- ◆ CacheSizingStatistics
- ◆ CarverHeapPages
- ◆ ConsoleLogMaxSize
- ◆ CollectStatistics
- ◆ DebuggingInformation
- ◆ DefaultNCharCollation
- ◆ FirstOption
- ◆ HeapsRelocatable
- ◆ HeapsLocked
- ◆ HeapsQuery
- ◆ HeapsCarver
- ◆ IsEccAvailable
- ◆ IsRsaAvailable
- ◆ LastConnectionProperty
- ◆ LastDatabaseProperty
- ◆ LastOption
- ◆ LastServerProperty
- ◆ MaxConnections
- ◆ MaxConcurrentReq
- ◆ NumLogicalProcessors
- ◆ NumLogicalProcessorsUsed
- ◆ NumPhysicalProcessors
- ◆ NumPhysicalProcessorsUsed
- ◆ QueryHeapPages

- ◆ RequestsReceived
- ◆ ServerName
- ◆ StartDBPermission

For more information about these properties, see [“Server-level properties” \[SQL Anywhere Database Administration Guide, page 481\]](#).

- **Database properties** The following database properties have been added in this release:

- ◆ AccentSensitive
- ◆ AlternateServerName
- ◆ EncryptionScope
- ◆ IOParallelism
- ◆ LockCount
- ◆ ReceivingTracingFrom
- ◆ SendingTracingTo
- ◆ SnapshotCount
- ◆ VersionStorePages

For more information about these properties, see [“Database-level properties” \[SQL Anywhere Database Administration Guide, page 492\]](#).

- **Performance monitor statistics properties** The following Performance Monitor statistics have been added in this release:

- ◆ Cache: Multi-Page Allocations
- ◆ Cache: Panics
- ◆ Cache: Scavenge Visited
- ◆ Cache: Scavenges
- ◆ Cache Pages: Allocated Structures
- ◆ Cache Pages: File
- ◆ Cache Pages: File Dirty
- ◆ Cache Pages: Free
- ◆ Comm: Requests Received
- ◆ Heaps: Carver
- ◆ Heaps: Query Processing
- ◆ Heaps: Relocatable Locked
- ◆ Heaps: Relocatable
- ◆ Mem Pages: Carver
- ◆ Mem Pages: Pinned Cursor
- ◆ Mem Pages: Query Processing
- ◆ Version Store Pages

For more information about these statistics, see [“Performance Monitor statistics” \[SQL Anywhere SQL User’s Guide, page 242\]](#).

- ◆ **Database connection enhancements**

- **Support for single or double quotes** Values in connection strings can now be enclosed in single or double quotes. This allows characters

such as spaces and semicolons to be used in connection string values. See [“Connection parameters passed as connection strings”](#) [*SQL Anywhere Database Administration Guide*, page 43] and [“Connection parameter tips”](#) [*SQL Anywhere Database Administration Guide*, page 67].

- **Connection strings now allow T, Y, F, and N as boolean values**
You can now specify T or Y to indicate true, and F or N to indicate false, when specifying connection parameters and protocol options in connection strings. See [“Connection parameters”](#) [*SQL Anywhere Database Administration Guide*, page 200].
- **Some connection strings and protocol options now accept values with k, m, and g suffixes** The following connection parameters and protocol options now accept k, m, and g as suffixes indicating kilobytes, megabytes, and gigabytes, respectively:
 - ◆ [“CommBufferSize connection parameter \[CBSIZE\]”](#) [*SQL Anywhere Database Administration Guide*, page 204]
 - ◆ [“CompressionThreshold connection parameter \[COMPTH\]”](#) [*SQL Anywhere Database Administration Guide*, page 208]
 - ◆ [“PrefetchBuffer connection parameter \[PBUF\]”](#) [*SQL Anywhere Database Administration Guide*, page 227]
 - ◆ [“LogMaxSize protocol option \[LSIZE\]”](#) [*SQL Anywhere Database Administration Guide*, page 243]
 - ◆ [“MaxRequestSize protocol option \[MAXSIZE\]”](#) [*SQL Anywhere Database Administration Guide*, page 245]
 - ◆ [“ReceiveBufferSize protocol option \[RCVBUFSZ\]”](#) [*SQL Anywhere Database Administration Guide*, page 246]
 - ◆ [“SendBufferSize protocol option \[SNDBUFSZ\]”](#) [*SQL Anywhere Database Administration Guide*, page 247]
- **AppInfo returns IP address for Windows clients** In previous releases, the AppInfo connection parameter only returned the IP address of the client computer on Unix and NetWare clients. The IP address is now returned for Windows clients as well. See [“AppInfo connection parameter \[APP\]”](#) [*SQL Anywhere Database Administration Guide*, page 201].
- **Auditing individual connections** The conn_auditing temporary database option allows you to enable or disable auditing for a specific connection when the option is set in a login procedure. The Auditing database property, and the Auditing and ConnAuditing properties have been added to help you obtain information about the auditing status of a database. See [“conn_auditing option \[database\]”](#) [*SQL Anywhere Database Administration Guide*, page 385].
- **RetryConnectionTimeout connection parameter** The RetryConnectionTimeout (RetryConnTO) connection parameter tells

the client library to retry the connection attempt, as long as the server is not found, for the specified period of time. See

[“RetryConnectionTimeout connection parameter \[RetryConnTO\]”](#)
[\[SQL Anywhere Database Administration Guide, page 230\]](#).

- ◆ **Support for parallel database backups** The SQL Anywhere database server now supports parallel backups for server-side image backups. Parallel database backups take advantage of physical I/O to perform read and write information in parallel, instead of sequentially, which improves performance. You can perform parallel backups in any of the following ways:
 - ◆ [“Backup utility \(dbbackup\)” \[SQL Anywhere Database Administration Guide, page 547\]](#)
 - ◆ [“BACKUP statement” \[SQL Anywhere Reference, page 318\]](#)
 - ◆ [“db_backup function” \[SQL Anywhere Programming Guide, page 434\]](#)
- ◆ **Tracking information on the last backup** A new column, LAST_BACKUP, has been added to the ISYSHISTORY system table to store information about the last backup. See [“SYSHISTORY system view” \[SQL Anywhere Reference, page 753\]](#).
- ◆ **Creating proxy ports for Windows CE** In previous releases of the software, you had to modify entries in the registry to configure Active Sync to use a proxy port for connecting to database on Windows CE devices. The Connect dialog for Interactive SQL, Sybase Central, and the SQL Anywhere Console utility now includes a Setup Windows CE Proxy Port tool that allows you to create proxy ports for connecting to databases on Windows CE devices without editing the registry. See [“Creating proxy ports for Windows CE devices” \[SQL Anywhere Database Administration Guide, page 933\]](#).
- ◆ **Directory access servers** You can now create a remote server that accesses the directory structure of the computer running the database server by creating a data access server. See [“Using a directory access server” \[SQL Anywhere SQL User’s Guide, page 727\]](#).
- ◆ **Checksums calculated automatically for critical database pages** The database server records checksums for critical database pages, regardless of whether checksums are enabled for the database. As a result, you may see warnings about checksum violations when you validate your database, even if the database does not have checksums enabled. Additionally, the database server shuts down with a fatal error when it tries to access a corrupt critical page. See [“Validating checksums” \[SQL Anywhere Database Administration Guide, page 732\]](#).

- ◆ **Applying multiple transaction logs at startup for recovery** By default, you must apply transaction logs individually, in the correct order when recovering a database. When the new -ad, -ar, and -as recovery options specified when starting the database server, you do not need to manually specify the order in which transaction logs are applied to the database. Because the database server and the database are running while the transaction logs are applied, the server's cache remains in a warm state, reducing total recovery time. See [“-ad database option” \[SQL Anywhere Database Administration Guide, page 190\]](#), [“-ar database option” \[SQL Anywhere Database Administration Guide, page 191\]](#), and [“-as database option” \[SQL Anywhere Database Administration Guide, page 191\]](#).
- ◆ **Index Consultant enhancements** The Index Consultant has been enhanced to improve recommendations with respect to clustered indexes, database and server states in the workload, and complete workload statistics reporting. It has been integrated into the Application Profiling tool.
- ◆ **Norwegian collation** 1252NOR has been added to support Norwegian. On Norwegian Windows systems, the database server chooses 1252NOR as the default collation for a new database if no collation is specified. See [“Supplied and recommended collations” \[SQL Anywhere Database Administration Guide, page 301\]](#).
- ◆ **UTF8BIN collation** The UTF8BIN collation has been added to offer improved sorting of binary data. This new collation replaces the UTF8 collation, which is now deprecated. See [“Supplied and recommended collations” \[SQL Anywhere Database Administration Guide, page 301\]](#).
- ◆ **Server Messages window enhancements** The following enhancements have been made to the Server Messages window:
 - **New right-click choices for the window title bar** On all supported Windows platforms (except Windows CE), when you right-click the title bar of the Server Messages window you can now choose About or Clear Message Area. Choosing About displays information about the database server, while choosing Clear Message Area erases all of the messages in the Server Messages window. Replicas of this window (the server output log, the Sybase Central Design Details window, and the SQL Anywhere Console utility) are not affected by the clearing action.
 - **Environment variables used by database server can be logged to Server Messages window** The -ze server option displays a list of the environment variables used by a database server in the Server Messages window. This feature is not available on NetWare or

Windows CE. See “-ze server option” [*SQL Anywhere Database Administration Guide*, page 184].

- **Server Messages window enhancements** By default, the Server Messages window minimizes once the database server starts. You can specify the -qn option if you do not want the Server Messages window to minimize once the database server is started. See “-qn server option” [*SQL Anywhere Database Administration Guide*, page 164].
- ◆ **Support for Veritas Cluster Server** SQL Anywhere now provides a Veritas Cluster Server agents. See “Using the SQL Anywhere Veritas Cluster Server agents” [*SQL Anywhere Database Administration Guide*, page 799].
- ◆ **New parameters for LDAP registration** The read_authdn and read_password parameters can be used to register the database server with LDAP if the database server is an Active Directory server. See “Connecting using an LDAP server” [*SQL Anywhere Database Administration Guide*, page 108].
- ◆ **Support for character length semantics in date formatting** Date format strings now use character length semantics to control the amount of text substituted for format specifier. For example, when using strings to format date strings such as MMM this used to imply the use of 3 bytes to store the month.
- ◆ **New SQL Flagger feature** The SQL Flagger provides information about SQL Anywhere-specific SQL constructs. The SQL Flagger feature is accessed using a set of system procedures and functions.
- ◆ **Support for IPv6** SQL Anywhere clients and database servers now support IPv6 on Windows XP, Windows 2003, and Linux. Servers running on these operating systems now listen on all available IPv4 and IPv6 addresses, and anywhere you can specify an IP address on the client or server (such as the HOST=, MYIP=, and BROADCAST= TCP parameters), you can now specify an IPv6 address. See “IPv6 support in SQL Anywhere” [*SQL Anywhere Database Administration Guide*, page 105].
- ◆ **Ability to track when a table was last updated** The database server now keeps track of the last time a table was updated. This is achieved using the new last_modified_at column in the SYSTAB system view. See “SYSTAB system view” [*SQL Anywhere Reference*, page 783].
- ◆ **JDBC driver now supports the SQL Server Native Client ODBC driver** The JDBC driver now checks if the ODBC driver is the SQL Server Native Client ODBC driver and appropriately sets the default result set type and other attributes.

- ◆ **SNMP traps when changing to another server during mirroring** The SNMP agent now sends a trap when it is connected to a server involved in mirroring, the connection drops, and a new connection is reestablished but to a *different* server.

This trap indicates that the original server went down, and the server that was acting as the mirror became the primary. See “Using traps” [[SQL Anywhere Database Administration Guide](#), page 678].

- ◆ **Changes to request logging** The request log is now stored in a comma-delimited text format, reducing it to roughly one third of its original size. Also, where possible, instead of a normal time entry, times are now recorded as either an equal sign (=), which means the same time as the previous entry in the log, or +*nnn*, where *nnn* is the number of milliseconds after the previous entry in the log. Additional information is now also recorded. For example, for queries, the isolation level, number of rows fetched, and cursor type are now recorded. For INSERT, UPDATE, and DELETE statements, the number of rows affected and number of triggers fired are now recorded.

The `sa_get_request_times` system procedure supports only the new request log format. However, the `tracetime` Perl script, *tracetime.pl*, processes both old and new request log formats. The `tracetime` script also performs faster on logs of the new format, noticeably so on large request logs.

Security

This section explains the enhancements made to SQL Anywhere to improve security.

- ◆ **RSA now included with SQL Anywhere** You no longer have to purchase a separate license to use RSA encryption. See “[Separately licensed components](#)” [[Introducing SQL Anywhere 10](#), page 13].
- ◆ **Enhancements to FIPS support** The following FIPS-related changes have been made to the database server:
 - The FIPS DLL has been renamed from *dbrsa10f.dll* to *dbfips10.dll* (similar name change for Unix).
 - The HASH function now accepts two new algorithms: SHA1_FIPS and SHA256_FIPS. These are the same as the SHA1 and SHA256 algorithms, but are the FIPS-validated Certicom versions.
 - If -FIPS is specified, the HASH function uses SHA1_FIPS. If SHA1 is specified, it uses SHA256_FIPS. If SHA256 is specified, it returns an error if MD5 is used (MD5 is not a FIPS algorithm), and password hashing uses the Certicom FIPS-validated SHA-1 code.

Also, the -FIPS option and FIPS functionality are now available on more platforms. For a list of supported platforms, see [components_platform_10.html#SA_SeparatelyLicensed](#), located in the *support* subdirectory of your SQL anywhere installation.

- ◆ **Kerberos authentication** SQL Anywhere now supports Kerberos authentication. Kerberos authentication lets you use your Kerberos credentials to connect to the database without specifying a user ID or password. See [“Using Kerberos authentication”](#) [*SQL Anywhere Database Administration Guide*, page 89].
- ◆ **New authorities added** The following authorities have been added:
 - **BACKUP authority** You can assign BACKUP authority to a user so that they can perform backups, instead of granting the user DBA authority. See [“GRANT statement”](#) [*SQL Anywhere Reference*, page 523] and [“Ways of making backups”](#) [*SQL Anywhere Database Administration Guide*, page 717].
 - **VALIDATE authority** A new authority for validation operations, VALIDATE, has been added. VALIDATE authority is required to perform the operations executed by the different VALIDATE statements, such as database, table, index, and checksum validation. See [“GRANT statement”](#) [*SQL Anywhere Reference*, page 523].
- ◆ **Disabling features for a database server** The -sf database server option lets you specify features, or groups of features, that are disabled for databases running on the database server. See [“-sf server option”](#) [*SQL Anywhere Database Administration Guide*, page 168].

The -sk server option lets you specify a key that can be used to enable disabled features when used with the secure_feature_key database option. You can also change the set of disabled features using the sa_server_option system procedure SecureFeatures property. See [“-sk server option”](#) [*SQL Anywhere Database Administration Guide*, page 172].

Database utilities

- ◆ **@filename can be reused for several utilities** Command parameter files can now be selectively parsed for the utility using the parameter file. The parsing is based on simple conditional directives placed in the parameter file. See [“Using conditional parsing in configuration files”](#) [*SQL Anywhere Database Administration Guide*, page 544].
- ◆ **Data Source utility (dbdsn) enhancements** The following options have been added to the dbdsn utility:

- **-dr** includes the DRIVER= parameter when you list the command that was used to create a data source. This allows you to recreate data sources so that they use a different version of the ODBC driver than the one included with the current version of the software.
- **-f** displays the name of the *.odbc.ini* file being used, and then list the DSNs in that file.
- **-ns** tells dbdsn not to search for the *.odbc.ini* file, but to use the existing environment variables to determine where the file should be. This is useful when the file specified by one or more of the environment variables does not exist, and a ODBC data source is being created.
- **-pe** encrypts the password field in the data source.

See “Data Source utility (dbdsn)” [*SQL Anywhere Database Administration Guide*, page 552].

- ◆ **Histogram utility (dbhist) enhancements** Sheets within the Excel output file created by dbhist are now named to reflect the column name they apply to, instead of Sheet1, Sheet2, and so on. See “Histogram utility (dbhist)” [*SQL Anywhere Database Administration Guide*, page 568].
- ◆ **Information utility (dbinfo) enhancements** The -u option now includes information about materialized views. See “Information utility (dbinfo)” [*SQL Anywhere Database Administration Guide*, page 571].
- ◆ **Initialization utility (dbinit) enhancements** The Initialization utility (dbinit) now supports the following new options:
 - **-a** uses accent sensitivity for UCA string comparisons
 - **-af** uses French accent sensitivity rules for UCA string comparisons.
 - **-dba** changes the user ID and/or password of the default DBA database user in a new database.
 - **-dbs** specifies the size of the database file.
 - **-ze** specifies the character set encoding for the CHAR data type.
 - **-zn** specifies the collation sequence for the NCHAR data type.

See “Initialization utility (dbinit)” [*SQL Anywhere Database Administration Guide*, page 573].
- ◆ **Log Transfer Manager (LTM) enhancements** The Log Transfer Manager (LTM) utility, also known as the Replication Agent, now supports identifiers up to 128 bytes for table, column, procedure, function, and parameter names when using the Replication Agent with Replication Server 15.0 and Open Server/Open Client 15.0. In earlier versions of the software, identifiers were limited to 30 bytes. See

“Identifiers in Replication Server” [[SQL Anywhere Database Administration Guide](#), page 897].

Timestamps in informational, warning, and error messages generated by dbltm now use the non-ambiguous ISO 8601 datetime format: {**I**|**W**|**E**} *yyyy-mm-dd hh:mm:ss message*.

- ◆ **Ping utility (dbping) enhancements** You can use the Ping utility (dbping) to obtain information about the performance of embedded SQL connections by specifying the -s or -st options. These options report statistics about the performance between the computer running dbping and the computer running the database server. See “[Testing embedded SQL connection performance](#)” [[SQL Anywhere Database Administration Guide](#), page 76].

The -pd option now lets you specify the name of the database you want to obtain the property value from. See “[Ping utility \(dbping\)](#)” [[SQL Anywhere Database Administration Guide](#), page 603].

- ◆ **Server Enumeration utility (dblocate) enhancements** The Server Enumeration utility (dblocate) now supports several new options to search for databases:
 - **-d** displays the server name and address, and a comma-separated list of all databases running on each server.
 - **-dn** displays the server name and address only if the server is running a database with the specified name.
 - **-dv** displays the server name and address, and lists all databases running on each server on a separate line.
 - **-p** displays servers using the specified TCP/IP port number.
 - **-s** displays servers with the specified name.
 - **-ss** displays server names that contain the specified substring.

See “[Server Enumeration utility \(dblocate\)](#)” [[SQL Anywhere Database Administration Guide](#), page 609].

- ◆ **Service utility (dbsvc) enhancements** The Service utility (dbsvc) supports the DBLTM service type, which allows you to manage services for the Log Transfer Manager, and the dbsln service type, which lets you manage services for the Listener utility.

The Service utility also supports the -o option, which allows you to log output from the utility to a file. See “[The Service utility](#)” [[SQL Anywhere Database Administration Guide](#), page 616].

- ◆ **New SQL Anywhere Broadcast Repeater utility (dbns10)** The SQL Anywhere Broadcast Repeater utility allows SQL Anywhere clients to

find SQL Anywhere database servers on other subnets and through firewalls, where UDP broadcasts wouldn't normally reach, without using the HOST parameter or LDAP. See [“The SQL Anywhere Broadcast Repeater utility”](#) [*SQL Anywhere Database Administration Guide*, page 624].

- ◆ **New SQL Anywhere Report Submission utility (dbsupport)** The new Support utility (dbsupport) provides the ability to submit error reports and statistics, the ability to query for updates (availability of EBFs), and the ability to check if previously submitted problems have been fixed. See [“SQL Anywhere Report Submission utility \(dbsupport\)”](#) [*SQL Anywhere Database Administration Guide*, page 630].

- ◆ **Unload utility (dbunload) enhancements** The following dbunload enhancements have been made:

- unprocessed statements are logged when dbunload encounters a failure. See [“Failed unloads”](#) [*SQL Anywhere Database Administration Guide*, page 646].
- support for binary data in unloaded tables.
- many internal enhancements have been made to improve performance for unloading databases

The following new options have been added:

- **-dc** recalculates the values for all computed columns in the database.
- **-g** initializes materialized views during reload.
- **-k** creates an auxiliary table for tracing support. Specifying this option populates the sa_diagnostic_auxiliary_catalog table. This option is useful when creating a tracing database.
- **-nl** creates a *reload.sql* file that includes LOAD TABLE and INPUT statements for each table, but no data.

See [“The Unload utility”](#) [*SQL Anywhere Database Administration Guide*, page 642].

- ◆ **Validation utility (dbvalid)** A new database validation option, -d, has been added. This option performs a database validation that includes checksum validation, checks for orphaned table pages and BLOBs, as well as a structural check. Indexes and data are not checked. See [“Validation utility \(dbvalid\)”](#) [*SQL Anywhere Database Administration Guide*, page 658].

Stored procedures and functions

Following are several new system procedures and functions, and new extensions to existing system procedures and functions.

- ◆ **Enhancements to all procedures and functions to support the DEFAULT clause** For procedures and user-defined functions, the value DEFAULT may be provided as an argument if the corresponding parameter was defined with a default value. In cases where the procedure has several parameters and the ones being defaulted are not all at the end, it may be easier to specify DEFAULT in the argument list than to use named parameters. Also, named parameters are not permitted in function calls.
- ◆ **New system procedures** The following system procedures have been added:
 - **sa_column_stats system procedure** The sa_column_stats system procedure returns string-related statistics about the specified column(s). See “sa_column_stats system procedure” [[SQL Anywhere Reference, page 837](#)].
 - **sa_conn_list system procedure** The sa_conn_list system procedure returns a connection ID. See “sa_conn_list system procedure” [[SQL Anywhere Reference, page 843](#)].
 - **sa_conn_options system procedure** The sa_conn_options system procedure returns property information for connection properties that correspond to database options. See “sa_conn_options system procedure” [[SQL Anywhere Reference, page 844](#)].
 - **sa_db_list system procedure** The sa_db_list system procedure returns a database ID. See “sa_db_list system procedure” [[SQL Anywhere Reference, page 849](#)].
 - **sa_describe_query system procedure** The sa_describe_query system procedure returns one row per column and describes the domain of the result expression and its nullability. This procedure is equivalent to performing the EXPRTYPE function on each column. See “sa_describe_query system procedure” [[SQL Anywhere Reference, page 850](#)].
 - **sa_refresh_materialized_views system procedure** The sa_refresh_materialized_views system procedure refreshes all materialized views in the database that are currently in an uninitialized state. See “sa_refresh_materialized_views system procedure” [[SQL Anywhere Reference, page 901](#)].
 - **sa_materialized_view_info system procedure** The sa_materialized_view_info system procedure returns information about a specified materialized view, such as its status and the owner of the view. See “sa_materialized_view_info system procedure” [[SQL Anywhere Reference, page 878](#)].

- **sa_remove_tracing_data system procedure** This procedure permanently deletes all record of a given logging session from the diagnostic tracing tables. See [“sa_remove_tracing_data system procedure” \[SQL Anywhere Reference, page 902\]](#).
- **sa_save_trace_data system procedure** This procedure saves data from temporary tracing tables to the base tables. See [“sa_save_trace_data system procedure” \[SQL Anywhere Reference, page 906\]](#).
- **sa_set_tracing_level system procedure** Sets the level of tracing data to generate for the database being profiled. See [“sa_set_tracing_level system procedure” \[SQL Anywhere Reference, page 916\]](#).
- **sa_snapshots** Returns a list of snapshots that are currently active for the database. See [“sa_snapshots system procedure” \[SQL Anywhere Reference, page 917\]](#).
- **sa_split_list system procedure** Takes a list of values and returns a result set containing that list. See [“sa_split_list system procedure” \[SQL Anywhere Reference, page 918\]](#).
- **sa_transactions** Returns a list of transactions that are currently running against a database. See [“sa_transactions system procedure” \[SQL Anywhere Reference, page 922\]](#).
- **sa_unload_cost_model and sa_load_cost_model system procedures** You can now unload the cost model from one database and load it into another database using the new system procedures `sa_unload_cost_model` and `sa_load_cost_model`, respectively. This eliminates repetitive, time-consuming recalibration activities when there is a large number of similar hardware installations. See [“sa_unload_cost_model system procedure” \[SQL Anywhere Reference, page 923\]](#) and [“sa_load_cost_model system procedure” \[SQL Anywhere Reference, page 871\]](#).
- ◆ **New functions** The following functions have been added:
 - **BIT_LENGTH function** Returns the number of bits stored in the array. See [“BIT_LENGTH function \[Bit array\]” \[SQL Anywhere Reference, page 104\]](#).
 - **BIT_SUBSTR function** Returns a sub-array of a bit array. See [“BIT_SUBSTR function \[Bit array\]” \[SQL Anywhere Reference, page 104\]](#).
 - **BIT_AND function** Takes two bit arrays and returns a bitwise AND-ing of its arguments using the following logic: for each bit compared, if both bits are 1, return 1; otherwise, return 0. See [“BIT_AND function \[Aggregate\]” \[SQL Anywhere Reference, page 105\]](#).

- **BIT_OR function** Takes two bit arrays and returns a bitwise OR-ing of its arguments using the following logic: for each bit compared, if either bit (or both) is 1, return 1; otherwise, return 0. See “[BIT_OR function \[Aggregate\]](#)” [*SQL Anywhere Reference*, page 107]
- **BIT_XOR function** Takes two bit arrays and returns a bitwise exclusive OR-ing of its arguments using the following logic: for each bit compared, if just one bit (but not both) is 1, return 1; otherwise, return 0. See “[BIT_XOR function \[Aggregate\]](#)” [*SQL Anywhere Reference*, page 108].
- **COUNT_SET_BITS function** Returns a count of the number of bits set to 1 (TRUE) in the array. See “[COUNT_SET_BITS function \[Bit array\]](#)” [*SQL Anywhere Reference*, page 125].
- **GET_BIT function** Returns the value (1 or 0) of a specified bit in a bit array. See “[GET_BIT function \[Bit array\]](#)” [*SQL Anywhere Reference*, page 155].
- **REVERSE function** This new function returns the reverse of a character expression. See “[REVERSE function \[String\]](#)” [*SQL Anywhere Reference*, page 213].
- **SET_BIT function** Sets the value of a specific bit in a bit array. See “[SET_BIT function \[Bit array\]](#)” [*SQL Anywhere Reference*, page 219].
- **SET_BITS function** Creates a bit array where specific bits, corresponding to values from a set of rows, are set to 1 (TRUE). See “[SET_BITS function \[Aggregate\]](#)” [*SQL Anywhere Reference*, page 220].
- **TRACED_PLAN function** Generates a graphical plan for a query using tracing data and information about optimizer conditions when the query was traced. See “[TRACED_PLAN function \[Miscellaneous\]](#)” [*SQL Anywhere Reference*, page 239].
- ◆ **Enhancements to various system procedures and functions** The following system procedures and functions have been enhanced as described:
 - **Enhancements to property functions** Property functions can now return LONG VARCHAR.
See:
 - ◆ “[CONNECTION_PROPERTY function \[System\]](#)” [*SQL Anywhere Reference*, page 117]
 - ◆ “[DB_PROPERTY function \[System\]](#)” [*SQL Anywhere Reference*, page 139]
 - ◆ “[PROPERTY function \[System\]](#)” [*SQL Anywhere Reference*, page 195]
 - **Enhancements to DB_EXTENDED_PROPERTY function** You can now use the DB_EXTENDED_PROPERTY function with the

NextScheduleTime database property to obtain the next scheduled execution time for an event. You can also use the function the return extended information about the CHAR character set. See [“DB_EXTENDED_PROPERTY function \[System\]” \[SQL Anywhere Reference, page 136\]](#).

- **New CONNECTION_EXTENDED_PROPERTY function** You can use the CONNECTION_EXTENDED_PROPERTY function to find out extended information for certain connection parameters. See [“CONNECTION_EXTENDED_PROPERTY function \[String\]” \[SQL Anywhere Reference, page 116\]](#).
- **sa_procedure_profile system procedure** The output from sa_procedure_profile system procedure can now be saved to a file, has new syntax, requires fewer parameters, and has new uses. See [“sa_procedure_profile system procedure” \[SQL Anywhere Reference, page 895\]](#).
- **sa_procedure_profile_summary system procedure** The sa_procedure_profile_summary system procedure now supports saving its output to file, has new syntax, accepts fewer parameters, and has new uses. See [“sa_procedure_profile_summary system procedure” \[SQL Anywhere Reference, page 898\]](#).
- **sa_server_option system procedure** The sa_server_option system procedure lets you change settings for the database server while it is still running. You can now change the following settings:
 - ◆ **CacheSizingStatistics property** Display cache information in the Server Messages window whenever the cache size changes.
 - ◆ **CollectStatistics property** Collect Performance Monitor statistics for the database server.
 - ◆ **ConsoleLogFile property** Specify the name of the output file where Server Messages window information is recorded.
 - ◆ **ConsoleLogMaxSize property** Specify the maximum size of the output file used to record Server Messages window information.
 - ◆ **DebuggingInformation property** Display diagnostic communication messages and other messages for troubleshooting purposes.
 - ◆ **IdleTimeout server option** Disconnect TCP/IP or SPX connections that have not submitted a request for the specified number of minutes.
 - ◆ **ProfileFilterConn property** Capture profiling information for a specific connection ID, without preventing other connections from using the database.
 - ◆ **RequestFilterDB property** You can use the sa_server_option system procedure to disable connections to a particular database.

- ◆ **RequestLogging property** The request log can now record blocking and unblocking events, plan information, procedures, and triggers.
- ◆ **RequestTiming property** Turning on request timing instructs the database server to maintain timing information for each request.

For more information, see “[sa_server_option system procedure](#)” [*SQL Anywhere Reference*, page 907].

- **Enhancement to xp_sendmail system procedure** The xp_sendmail system procedure now supports attachments when sending mail using SMTP, using the new include_file parameter. In addition, xp_sendmail supports MIME content when using SMTP mail, using the new content_type parameter. See “[xp_sendmail system procedure](#)” [*SQL Anywhere Reference*, page 942].
- **sa_conn_info system procedure now returns AppInfo connection property value** The sa_conn_info system procedure now returns information about the client process, such as the IP address of the client computer, the operating system it is running on, and so on. The information returned is the same information as would be returned for the AppInfo connection property. See “[sa_conn_info system procedure](#)” [*SQL Anywhere Reference*, page 841].
- **sa_performance_diagnostics returns more information** The sa_performance_diagnostics system procedure now returns the LockCount and SnapshotCount when you use snapshot isolation. See “[sa_performance_diagnostics system procedure](#)” [*SQL Anywhere Reference*, page 890].
- **Enhancement to the HASH function** The HASH function now accepts the following new algorithms: SHA256, SHA1_FIPS, and SHA256_FIPS. The FIPS related algorithms are only for use on systems that use FIPS-certified software. See “[HASH function \[String\]](#)” [*SQL Anywhere Reference*, page 159].
- **COMPRESS and DECOMPRESS functions support new algorithm** The gzip algorithm is now available to compress and decompress a string in a function. See “[COMPRESS function \[String\]](#)” [*SQL Anywhere Reference*, page 114], and “[DECOMPRESS function \[String\]](#)” [*SQL Anywhere Reference*, page 139].

SQL statements

Following are several new SQL statements, and new extensions to existing SQL statement syntax. These new features are in addition to statement changes listed in the previous feature sections of this document.

- ◆ **SQL statements to support materialized views** The following SQL statements have been added, or have had their syntax and functionality extended, to support materialized views:

- ◆ [“ALTER MATERIALIZED VIEW statement” \[SQL Anywhere Reference, page 283\]](#)
- ◆ [“COMMENT statement” \[SQL Anywhere Reference, page 340\]](#)
- ◆ [“CREATE INDEX statement” \[SQL Anywhere Reference, page 382\]](#)
- ◆ [“CREATE MATERIALIZED VIEW statement” \[SQL Anywhere Reference, page 388\]](#)
- ◆ [“DROP statement” \[SQL Anywhere Reference, page 475\]](#)
- ◆ [“REFRESH MATERIALIZED VIEW statement” \[SQL Anywhere Reference, page 596\]](#)
- ◆ [“VALIDATE statement” \[SQL Anywhere Reference, page 687\]](#)
- ◆ [“VALIDATE INDEX statement” \[SQL Anywhere Reference, page 690\]](#)

A new `OPTION` clause in the `SELECT` statement can be used to override the `materialized_view_optimization` database option. See [“SELECT statement” \[SQL Anywhere Reference, page 623\]](#).

- ◆ **New SQL statements to support diagnostic tracing and application profiling** The new SQL statements to support the Application Profiling feature are listed below:

- ◆ [“ATTACH TRACING statement” \[SQL Anywhere Reference, page 316\]](#)
- ◆ [“DETACH TRACING statement” \[SQL Anywhere Reference, page 473\]](#)
- ◆ [“REFRESH TRACING statement” \[SQL Anywhere Reference, page 598\]](#)

- ◆ **New VALIDATE DATABASE statement** You can now validate the database using the `VALIDATE DATABASE` statement. See [“VALIDATE statement” \[SQL Anywhere Reference, page 687\]](#).

- ◆ **New ALTER STATISTICS statement** You can now control whether column statistics are automatically using the `ALTER STATISTICS` statement. You can still force an update of statistics on columns where automatic updating has been disabled, using an explicit `CREATE STATISTICS` or `DROP STATISTICS` statement. See [“ALTER STATISTICS statement” \[SQL Anywhere Reference, page 298\]](#).

- ◆ **ALTER TABLE and CREATE TABLE statement enhancements** You now have finer control over what constitutes a match between a foreign key in a referencing table, and the primary key in the referenced table using the `MATCH` clause. You are also able to declare a foreign key as unique, thereby eliminating the need to declare uniqueness separately. See [“CREATE TABLE statement” \[SQL Anywhere Reference, page 427\]](#) and [“ALTER TABLE statement” \[SQL Anywhere Reference, page 303\]](#).

- ◆ **New CALIBRATE PARALLEL READ clause for the ALTER DATABASE statement** Use the new CALIBRATE PARALLEL READ clause of the ALTER DATABASE statement to detect hardware capable of parallel input and output. You can retrieve the calibration result for a dbspace by querying the new IOParallelism extended database property using the DB_EXTENDED_PROPERTY function. See [“ALTER DATABASE statement” \[SQL Anywhere Reference, page 271\]](#), and [“Database-level properties” \[SQL Anywhere Database Administration Guide, page 492\]](#).
- ◆ **New PRIMARY KEY ON clause for COMMENT statement** You can now create remarks for primary keys using the PRIMARY KEY ON clause of the COMMENT statement. See [“COMMENT statement” \[SQL Anywhere Reference, page 340\]](#).
- ◆ **CREATE ENCRYPTED FILE statement enhancement to change encryption keys** Using extensions to the CREATE ENCRYPTED FILE statement, you can now change the encryption key used to encrypt a database, transaction log, or dbspace without unloading and reloading the database. If the database is not encrypted, but table encryption is enabled, you can use the CREATE ENCRYPTED FILE statement to change the key used for table encryption. See [“CREATE ENCRYPTED FILE statement” \[SQL Anywhere Reference, page 362\]](#).
- ◆ **CREATE DATABASE statement enhancements** Three new clauses, ENCODING, NCHAR COLLATION, and ACCENT, have been added for improved handling of character sets. For more information on these clauses, see [“CREATE DATABASE statement” \[SQL Anywhere Reference, page 349\]](#).
- ◆ **SELECT statement enhancements** The FOR UPDATE clause, used in updating and deleting rows through a cursor, has been extended to allow columns lists.

The READPAST table hint has been added, which directs the database server to ignore locked rows.

The FORCE OPTIMIZATION clause has been added, which directs the database server to perform optimization on a query, even if the query qualifies for bypassing cost-based optimization.

☞ See [“SELECT statement” \[SQL Anywhere Reference, page 623\]](#).
- ◆ **LOAD TABLE and UNLOAD TABLE statement enhancements** The STRIP clause for the LOAD TABLE statement now accepts options that allow you to control whether leading blanks are stripped from unquoted values before they are inserted. Additional STRIP options let you fine tune how the data is stripped.

The LOAD TABLE statement has also been extended to support the COMMENTS INTRODUCED BY option. This option allows you to specify the string used to identify comments in the input data. Any lines in the input that begin with the specified string are ignored during the load operation.

Both the LOAD TABLE and UNLOAD TABLE statements have been extended to support the following options:

- **ENCODING option** Used to specify the encoding to use when loading or unloading data.
- **ROW DELIMITED BY option** Used to specify the string that indicates the end of an input record when bulk loading or unloading data.
- **QUOTE option** Similar to the QUOTE option for the OUTPUT statement in Interactive SQL. See “[OUTPUT statement \[Interactive SQL\]](#)” [*SQL Anywhere Reference*, page 577].

See “[LOAD TABLE statement](#)” [*SQL Anywhere Reference*, page 558], and “[UNLOAD TABLE statement](#)” [*SQL Anywhere Reference*, page 673].

- ◆ **VALIDATE INDEX statement enhancements** The syntax for VALIDATE INDEX has been enhanced to support index specifications. See “[VALIDATE INDEX statement](#)” [*SQL Anywhere Reference*, page 690].
- ◆ **Enhancements to the ALTER INDEX statement to rename primary keys** You can now rename primary keys using the ALTER INDEX statement. See “[ALTER INDEX statement](#)” [*SQL Anywhere Reference*, page 281].
- ◆ **New CONTINUE statement** Use this statement to restart a loop. Statements in the loop following the CONTINUE statement are skipped. See “[CONTINUE statement \[T-SQL\]](#)” [*SQL Anywhere Reference*, page 348].
- ◆ **New BREAK statement [T-SQL]** Use this statement to leave a compound statement or loop. See “[BREAK statement \[T-SQL\]](#)” [*SQL Anywhere Reference*, page 330].
- ◆ **Enhancement to the INSERT statement control updating default values during an INSERT** You can control whether default values are updated during an INSERT when a row already exists using the DEFAULTS ON | OFF clause. This new capability does not extend to the following default fields: DEFAULT TIMESTAMP, DEFAULT UTC TIMESTAMP, and DEFAULT LAST USER; these fields are always updated. See “[INSERT statement](#)” [*SQL Anywhere Reference*, page 548].
- ◆ **Enhancement to the DELETE statement to support an ORDER BY clause** The DELETE statement now supports the ORDER BY clause,

which allows you to specify the order in which rows are deleted from the database. See [“DELETE statement” \[SQL Anywhere Reference, page 461\]](#).

- ◆ **Enhancements to the START DATABASE statement** The START DATABASE statement now returns a wider range of error messages when the statement fails to indicate the reason why the database failed to start. As well, the START DATABASE clauses can now be specified in any order. See [“START DATABASE statement” \[SQL Anywhere Reference, page 650\]](#).
- ◆ **Enhancements to the CREATE DATABASE statement** The DATABASE SIZE clause has been added so you can specify the minimal initial size of a database. The DBA USER and DBA PASSWORD clauses allows you to specify the initial DBA user ID and password for the new database. See [“CREATE DATABASE statement” \[SQL Anywhere Reference, page 349\]](#).
- ◆ **Enhancement to the MESSAGE statement to support logging only to event or system log** In addition to being able to turn on or off logging, you can also specify whether to log only to the Event or System log. Syntax for the MESSAGE statement has been extended to allow the optional clause [EVENT | SYSTEM] in within the TO LOG clause. For example, TO EVENT LOG results in logging only to the Event log. See [“MESSAGE statement” \[SQL Anywhere Reference, page 570\]](#).
- ◆ **Support for temporary stored procedures** You can now create temporary stored procedures using an extension to the CREATE PROCEDURE statement. Temporary stored procedures are visible only by the connection that created them, and are automatically dropped when the connection is dropped. See [“CREATE PROCEDURE statement” \[SQL Anywhere Reference, page 391\]](#).
- ◆ **Support for local temporary tables** You can now create local temporary tables using the CREATE LOCAL TEMPORARY TABLE statement. Local temporary tables created this way are dropped when the connection closes. See [“CREATE LOCAL TEMPORARY TABLE statement” \[SQL Anywhere Reference, page 386\]](#).
- ◆ **Enhancements to temporary tables** You can now create global temporary tables whose data can be shared by all connections to a database, using the SHARE BY ALL clause of the CREATE GLOBAL TEMPORARY TABLE statement. See [“CREATE TABLE statement” \[SQL Anywhere Reference, page 427\]](#).

Data types

- ◆ **Support for character-length semantics for CHAR and VARCHAR**

data types When you specify a CHAR or VARCHAR column, you can now use character-length semantics. Character-length semantics allow you to express the length in characters, instead of bytes. See “[CHAR data type](#)” [*SQL Anywhere Reference*, page 50], and “[VARCHAR data type](#)” [*SQL Anywhere Reference*, page 54].

- ◆ **Support for bit array data types** SQL Anywhere now supports the VARBIT and LONG VARBIT data types. These data types are used to store bit arrays. See “[Bit array data types](#)” [*SQL Anywhere Reference*, page 66].

The following functions have been added for use with bit array data types:

- ◆ “[BIT_LENGTH function \[Bit array\]](#)” [*SQL Anywhere Reference*, page 104]
- ◆ “[BIT_SUBSTR function \[Bit array\]](#)” [*SQL Anywhere Reference*, page 104]
- ◆ “[BIT_AND function \[Aggregate\]](#)” [*SQL Anywhere Reference*, page 105]
- ◆ “[BIT_OR function \[Aggregate\]](#)” [*SQL Anywhere Reference*, page 107]
- ◆ “[BIT_XOR function \[Aggregate\]](#)” [*SQL Anywhere Reference*, page 108]
- ◆ “[COUNT_SET_BITS function \[Bit array\]](#)” [*SQL Anywhere Reference*, page 125]
- ◆ “[GET_BIT function \[Bit array\]](#)” [*SQL Anywhere Reference*, page 155]
- ◆ “[SET_BIT function \[Bit array\]](#)” [*SQL Anywhere Reference*, page 219]
- ◆ “[SET_BITS function \[Aggregate\]](#)” [*SQL Anywhere Reference*, page 220]

Programming interfaces

- ◆ **ADO.NET 2.0 support** The ADO.NET driver has been updated to support version 2.0 of the .NET framework. Several new classes and methods have been added as part of this support. See “[SQL Anywhere .NET Data Provider API Reference](#)” [*SQL Anywhere Programming Guide*, page 151].
- ◆ **SQL Anywhere Explorer** The SQL Anywhere Explorer lets you connect to SQL Anywhere databases from within Visual Studio .NET. In addition, you can open Sybase Central and Interactive SQL directly from Visual Studio .NET. See “[Working with data connections in Visual Studio .NET](#)” [*SQL Anywhere Programming Guide*, page 16].
- ◆ **Support for PreparedStatement.addBatch method** The iAnywhere JDBC driver now supports the PreparedStatement.addBatch method. This method is useful for performing batched (or wide) inserts.
- ◆ **JDBC driver (bridge) now supports JDBC 3.0 calls**
[under construction]

◆ **Support for SQL_GUID added to ODBC/OLE DB drivers**

[under construction] In version 8.0.2, unique identifier support was added to Adaptive Server Anywhere. A column can be typed as a “uniqueidentifier”, a 16-byte binary value. Previously, a uniqueidentifier column was typed as SQL_BINARY. Support specifically for uniqueidentifier columns has now been added to the ASA ODBC driver. A uniqueidentifier column will now and can now be typed as SQL_GUID.

Support for GUID escape sequences added to the iAnywhere ODBC driver

[under construction]

Support for GUID escape sequences has been added to the iAnywhere ASA ODBC driver. GUID escape sequences may be used in SQL statements.

- ◆ **ODBC message callbacks are now per-connection** ODBC has supported message callbacks since Adaptive Server Anywhere version 9.0.0, but messages for all connections came to a single callback function. As of version 9.0.2, when you designate a message callback function, it applies only to a single connection. This is consistent with how DBLIB works. All messages now funnel through a single function in the ODBC driver, which filters the messages by connection, and only calls the connection’s callback function for those connections that have one.

- ◆ **New functions added to the SQL Anywhere PHP module** The following new functions have been added to the SQL Anywhere PHP module:

- ◆ `sqlanywhere_execute`
- ◆ `sqlanywhere_error`
- ◆ `sqlanywhere_errorcode`
- ◆ `sqlanywhere_insert_id`

In addition, two new options have been added to the `sqlanywhere_set_option` function: `verbose_errors` and `row_counts`. See “SQL Anywhere PHP API reference” [[SQL Anywhere Programming Guide, page 489](#)].

- ◆ **Enhancements to `db_locate_servers_ex` function** The `db_locate_servers_ex` function supports two new flags: `DB_LOOKUP_FLAG_ADDRESS_INCLUDES_PORT`, which returns the TCP/IP port number in the `a_server_address` structure passed to the callback function, and `DB_LOOKUP_FLAG_DATABASES`, which indicates that the callback function is called once for each database or database server that is found. See “`db_locate_servers_ex` function” [[SQL Anywhere Programming Guide, page 444](#)].

New ODBC Driver Manager and ODBC driver enhancements

- ◆ **Perl DBD::ASAny driver for the Perl DBI module renamed to Anywhere databases from Perl scripts** [under construction]
- ◆ **ODBC driver manager enhancements** The ODBC Driver Manager now supports: all ODBC calls, wide CHAR entry points, tracing of connections. In addition, the ODBC Driver Manager is now able to switch between a non-threaded or threaded SQL Anywhere driver.
- ◆ **ODBC Driver Manager can now be used by both threaded and non-threaded applications** The ODBC Driver Manager can now be used by both threaded and non-threaded applications.

Deployment

- ◆ **Deployment wizard** The Deployment wizard has been added for creating deployments of SQL Anywhere for Windows. The Deployment wizard can be used to create both Microsoft Windows Installer package files and Microsoft Windows Installer Merge Module files. The Deployment wizard is not installed by default, but can be installed at any time. See “[Using the Deployment wizard](#)” [*SQL Anywhere Programming Guide*, page 656].

Unix/Linux enhancements

- ◆ **New ODBC driver manager that can be used on Unix platforms** The libdbodm10 shared object can now be used on Unix platforms as the ODBC Driver Manager. Applications using the iAnywhere ODBC Driver Manager must restrict their ODBC reliance to version 3.0 and above.
- ◆ **-uf server option** The -uf option allows you to specify the action the database server takes when a fatal error occurs on Unix. See “[-uf server option](#)” [*SQL Anywhere Database Administration Guide*, page 176].
- ◆ **Service utility now supported on Linux** The Service utility is now available for use on Linux to create, delete, list, start, and stop SQL Anywhere services.
- ◆ **Additional samples supported on Unix** The following SQL Anywhere samples are now supported on Unix:
 - ◆ dbtools
 - ◆ diskfull
 - ◆ http
 - ◆ oem_string
 - ◆ performancefetch
 - ◆ performanceinsert

Web services

- ◆ **HTTP server supports keep-alive** The database server now supports the keep-alive option when requested by HTTP clients. Instead of closing

a connection after each request, an HTTP connection can be kept open after each request and response, so that multiple requests can be executed on the same connection.

- ◆ **Web server is HTTP 1.1 compliant** For HTTP 1.1 compliance, the web server now accepts the following items:
 - ◆ pipelining of HTTP requests, enabling multiple HTTP requests such as GET and HEAD to be processed simultaneously
 - ◆ Keep-Alive request-header field, to keep a connection available for reuse (useful when pipelining)
 - ◆ absolute URIs (previously only relative URIs were supported)
 - ◆ 100-continue request-header field, enabling a client to determine if the server would accept a request (based on the request headers) before the client sends the entire request body.
 - ◆ quality values in the Accept-Charset request-header field (these values were previously ignored)
- ◆ **HTTP clients can now add or modify request headers** An HTTP client stored procedure can now add or modify its HTTP request headers using a new header clause in the CREATE PROCEDURE and ALTER PROCEDURE statements.
- ◆ **New HttpServiceName connection property** A new connection property, HttpServiceName, has been added to enable a web application to determine its service name origin. The property is useful for error reporting and control flow.
- ◆ **webservice_namespace_host option added** The webservice_namespace_host option specifies the hostname to be used within the generation of the WSDL (Webservices Description Language exported by the DISH service). The option is useful when developing web service client applications that target a host other than SQL Anywhere. See “[webservice_namespace_host option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 455].
- ◆ **sa_set_http_option enhancements** You can now use the sa_set_http_option system procedure to specify the character set used in the HTTP response. See “[sa_set_http_option system procedure](#)” [*SQL Anywhere Reference*, page 915].
- ◆ **sa_set_soap_header system procedure** You can use the sa_set_soap_header system procedure to set the response headers for SOAP services. See “[sa_set_soap_header system procedure](#)” [*SQL Anywhere Reference*, page 916].

- ◆ **SOAP_HEADER function** You can use the SOAP_HEADER function to get request headers for SOAP services. See [“SOAP_HEADER function \[String\]” \[SQL Anywhere Reference, page 223\]](#).
- ◆ **HTTP-related enhancements** The following HTTP-related enhancements are now implemented:
 - ◆ Support for HTTP string memory pooling
 - ◆ Client chunk mode
 - ◆ HTTP connections can create an HTTP session to maintain state between HTTP requests.

Miscellaneous

- ◆ **Indexing enhancements** The following enhancements have been made to indexing in this release:
 - **Support for index sharing** When you create a primary key, secondary key, foreign key, or unique constraint, you now create a logical index that points to a physical index (an actual indexing structure on disk). The database server automatically determines whether a new physical index is required to satisfy the logical index. This model allows the sharing of physical indexes and prevents the creation and maintenance of duplicate physical indexes, which wastes disk space. See [“Index sharing using logical indexes” \[SQL Anywhere SQL User's Guide, page 567\]](#).
 - **Improved storage of index information** There are several improvements to how index information is organized in the database. For example, the list of all indexes, including primary and foreign key indexes, are now stored in a single system table, ISYSIDX. Three new system tables, ISYSPHYSIDX, ISYSIDXCOL, and ISYSFKEY provide additional information about the indexes listed in ISYSIDX. See:
 - ◆ [“Indexes in the system tables” \[SQL Anywhere SQL User's Guide, page 98\]](#)
 - ◆ [“SYSIDX system view” \[SQL Anywhere Reference, page 755\]](#)
 - ◆ [“SYSPHYSIDX system view” \[SQL Anywhere Reference, page 765\]](#)
 - ◆ [“SYSIDXCOL system view” \[SQL Anywhere Reference, page 756\]](#)
 - ◆ [“SYSFKEY system view” \[SQL Anywhere Reference, page 751\]](#)
- ◆ **New outer join elimination rewrite optimization** Outer joins are eliminated from the query before execution if the resulting query is semantically equivalent to the original query. See [“Semantic query transformations” \[SQL Anywhere SQL User's Guide, page 492\]](#).

Version support

This section includes changes to the version numbers of supported and unsupported third-party components.

- ◆ **Support for version 8 and earlier databases** When using Sybase Central, Interactive SQL, or the SQL Anywhere Console utility, databases created with versions of the software older than Adaptive Server Anywhere 8.0.0 are no longer supported. This includes databases that were created with older software and then upgraded using newer software. Attempting to load such databases will result in an error on database startup. You can connect to these databases from Sybase Central to unload them into a new version 10 database. See [“Upgrading SQL Anywhere” on page 334](#).
- ◆ **jConnect version 5.5 and 6.0 supported** SQL Anywhere now supports jConnect versions 5.5 and 6.0 (version 5.5 was supported in 9.0.2 as well) for connecting to the database server.
- ◆ **Pentium class processor required for running the SQL Anywhere database server** The SQL Anywhere database server will not start on a computer unless it has a Pentium-class processor.
- ◆ **New files for jConnect** In addition to the changes mentioned, the following file and folder have been added:

New file name
Shared\jConnect-6_0
Shared\jConnect-6_0\jconn3.jar

Behavior changes

Following is a list of changes to SQL Anywhere databases introduced in version 10.0.0.

- ◆ **Adaptive Server Anywhere renamed** In version 10.0.0, Adaptive Server Anywhere has been renamed SQL Anywhere.
- ◆ **Upgrade changes** The Upgrade Database wizard, the Upgrade utility (dbupgrad), and ALTER DATABASE UPGRADE statement cannot be used to upgrade version 9.0.2 and earlier databases to version 10. To upgrade older databases to version 10, you must rebuild the database by performing an unload and reload. See [“Upgrading SQL Anywhere” on page 334](#).

- ◆ **Password changes** In newly-created databases, all passwords are case-sensitive, regardless of the case-sensitivity of the database. The default DBA password for new databases is **sql**.

When you upgrade an existing database, the case sensitivity of the password is determined as follows:

- If the password was originally entered in a case-insensitive database, the password remains case-insensitive.
- If the password was originally entered in a case-sensitive database, uppercase and mixed case passwords remain case-sensitive. However, if the password was entered in all lowercase, then the password becomes *case-insensitive*.
- Changes to existing passwords and new passwords are case-sensitive.

The database server now uses SHA256 to hash passwords. The old (proprietary) hashing algorithm is still supported for passwords reloaded from old databases, but all new passwords will use SHA256.

Passwords are now stored in UTF-8, so they continue to work if the database is reloaded into a database with a different character set.

In previous releases when connecting from embedded SQL, it was possible to connect to a database with DBA permission and then successfully make a second connection to the same database for any user without specifying the password. Now the password must be specified on every connection.

- ◆ **Case of return values for properties** Server properties (returned by the PROPERTY function) that returned YES or NO in previous releases now return Yes or No. Database properties (returned by the DB_PROPERTY function) and connection properties (returned by the CONNECTION_PROPERTY function) that returned ON or OFF in previous releases now return On or Off. This change may affect case-sensitive databases or applications that use case-sensitive string comparisons.
- ◆ **Connections to database servers with long names** On Windows and Unix, version 9.0.2 and earlier clients cannot connect to version 10.0.0 and later database servers with names longer than 40 bytes.
- ◆ **Changes to system procedures and functions** Following is a list of changes to system procedures and functions:
 - **Several system procedures have been made internal** The external system procedures `xp_read_file`, `xp_write_file`, `xp_sprintf`, `xp_scanf`, and `xp_cmdshell` are now internal system procedures.

- **sa_validate system procedure** The sa_validate system procedure now requires VALIDATE authority. See “sa_validate system procedure” [[SQL Anywhere Reference](#), page 923].
- **sa_reset_identity system procedure** The table-name parameter is now required. Additionally, if the owner-name parameter is not specified, the table-name parameter must uniquely identify a table in the database. See “sa_reset_identity system procedure” [[SQL Anywhere Reference](#), page 903].
- **sa_locks system procedure** The output of the sa_locks system procedure has been changed to return additional information, including the connection ID, the user ID, the table name, the lock class, and lock duration. See “sa_locks system procedure” [[SQL Anywhere Reference](#), page 872].
- **RAND function** Previously, each connection was seeded with the same value so that the RAND function would return identical sequences for each connection. Now, each connection will be uniquely seeded so that each connection will see a different random sequence. See “RAND function [Numeric]” [[SQL Anywhere Reference](#), page 197].
- **DB_CALLBACK_START and DB_CALLBACK_FINISH callback functions** The DB_CALLBACK_START and DB_CALLBACK_FINISH callback functions are now supported on all platforms (previously they were only supported on Windows platforms). See “db_register_a_callback function” [[SQL Anywhere Programming Guide](#), page 445].
- ◆ **Scattered reads no longer used for files specified using a UNC name** Scattered reads are now longer used for files on remote computers, or for files specified using a UNC name such as `\\mymachine\myshare\mydb.db`. See “Use an appropriate page size” [[SQL Anywhere SQL User's Guide](#), page 218].
- ◆ **Column ordering in primary and foreign key constraints** When creating primary key constraint, you can specify any order to the columns, regardless of the order in which columns appear in the table. Also, you can now create foreign keys that have a column order different from the primary key to which they refer, provided you specify the mapping between the foreign key columns and primary key columns.
- ◆ **Duplicate column names no longer allowed in indexes** Previously, duplicate references to columns in an index were allowed, except for primary key, foreign key, and unique constraint specifications. Now, the behavior is consistent across all types of indexes; specifying duplicate column names returns an error. Additionally, if an older database

contains an index with duplicate column references, the dbunload utility drops the duplicate columns from the index when generating reload.sql.

- ◆ **Encryption database property** Executing `SELECT DB_PROPERTY('Encryption')` may now return a value other than None, even when the database is not encrypted. This occurs when table encryption is enabled for the database. If you have an application that executes this command as a method for checking whether the database is encrypted, use `SELECT DB_PROPERTY('EncryptionScope')`, instead. See [“Understanding database properties” \[SQL Anywhere Database Administration Guide, page 458\]](#).
- ◆ **Change in syntax for starting HTTPS using FIPS** Previously, you would specify `-xs HTTPS_FIPS(. . .)`. Now, you must specify `-xs HTTPS(FIPS=yes; . . .)`. The former syntax is still supported, but is deprecated.
- ◆ **Maximum user ID length is 128 bytes** In previous releases, when a statement required a user ID, the database server truncated user IDs longer than 128 bytes before using them in database server. If the `string_rtruncation` option was set, a truncation error was returned. The database server now returns an error if you specify a user ID that is longer than 128 bytes, regardless of the setting of the `string_rtruncation` option. See [“Identifiers” \[SQL Anywhere Reference, page 8\]](#).
- ◆ **Maximum length for server names** The maximum length of database server names has been increased from 40 bytes to 250 bytes on TCP/IP and shared memory connections. See [“-n server option” \[SQL Anywhere Database Administration Guide, page 159\]](#).
- ◆ **Restrictions removed on user IDs, passwords, encryption keys, server names, and database names** In previous releases of the software user IDs, passwords, encryption keys, server names, and database names could not contain the following:
 - names that begin with white space or single or double quotes
 - names that end with white space
 - names that contain semicolonsThese restrictions have been removed in this release.
- ◆ **Changes to acceptable characters in identifiers** Double quotes and backslashes are no longer permitted in identifiers.
- ◆ **Changes to connection strings** For ODBC and OLE DB connections, the precedence of where a connection parameter is found is now: connection string, SQLCONNECT environment variable, data

source. Previously, in ODBC and OLE DB the data source had higher precedence than SQLCONNECT.

◆ **Empty value connection parameters now treated as unspecified**

For all APIs, connection parameters that are specified with empty values are treated as though the parameter was not specified. In previous releases, an empty value was treated as unspecified or as an empty string, depending on the location it was specified in and the API being used.

◆ **Leading and trailing spaces no longer ignored in connection strings**

Previously, spaces surrounding a parameter value in a connection string were ignored. For example, 'uid=DBA' was equivalent to 'uid= DBA '. Now, however, these spaces are no longer ignored since user IDs and passwords can contain significant leading and trailing spaces.

◆ **Transaction log cannot be turned off while auditing is on**

In previous versions of the software, you could stop using the transaction log for a database that had auditing turned on. Now, you must use a transaction log if auditing is turned on for a database. You must turn auditing off if you want to stop using the transaction log.

◆ **Databases with auditing turned on cannot be started in read-only mode**

In previous versions of the software, you could start databases in read-only mode with auditing turned on. Now, databases with auditing on cannot be started in read-only mode.

◆ **Precision of signed BIGINT columns now 19 instead of 20**

Previously, when an ODBC application described a *signed* BIGINT column using SQL_BIGINT, a precision value of 20 was returned, which was incorrect. Now, a value of 19 is returned. You need to change any applications that relied on the previous (incorrect) value.

◆ **Java VM enhancements**

SQL Anywhere no longer offers the Java option as a separately licensable component. Java in the database now uses an external VM to run your Java code instead of using an internal VM. As a result, you can now use any Java VM you want and you are no longer restricted to particular JDK versions or Java targets. Newly-initialized databases are always Java enabled.

This results in the following changes:

• **Unsupported database options**

The following options are no longer supported in SQL Anywhere:

- ◆ describe_java_format
- ◆ java_heap_size
- ◆ java_namespace_size
- ◆ java_page_buffer_size

- ◆ java_input_output
- ◆ return_java_as_string
- **New database options for Java support** The following database options have been added:
 - ◆ “java_location option [database]” [*SQL Anywhere Database Administration Guide*, page 407]
 - ◆ “java_main_userid option [database]” [*SQL Anywhere Database Administration Guide*, page 407]
- **Unsupported properties** Support has been removed for the following properties:
 - ◆ Database properties:
 - ◆ JDKVersion
 - ◆ JavaHeapSize
 - ◆ JavaNSSize
 - ◆ Database server properties:
 - ◆ IsJavaAvailable
 - ◆ JavaGlobFix
 - ◆ Connection properties:
 - ◆ JavaHeapSize
 - ◆ java_input_output
- **New JavaVM property** The JavaVM database property returns the name of the Java VM the database server uses to execute Java in the database.
- **Unsupported compatibility view columns** The following columns are no longer available in the system compatibility views:
 - ◆ SYSINFO.classes_version
 - ◆ SYSJAVACLASS.replaced_by
 - ◆ SYSJAVACLASS.type_id
- **Java options deprecated for database utilities** The following database utility options have been deprecated:
 - ◆ Initialization utility (dbinit): -ja, -jdk
 - ◆ Unload utility (dbunload): -jr
 - ◆ Upgrade utility (dbupgrad): -ja, -jdk, -jr, -j
- **Java support removed from CREATE DATABASE and ALTER DATABASE statements** The CREATE DATABASE statement no longer supports the JAVA ON | OFF and JDK version clauses, while the ALTER DATABASE statement no longer supports the REMOVE JAVA clause.
- **New Java files** In addition to the changes mentioned, the following file has been added:

`java\sajvm.jar`

- ◆ **Ping utility (dbping)** Previously, the Ping utility (dbping) reported an error if the database server returned NULL for a property value. Now, dbping prints NULL when a property value is unknown and exits with a success return code. You can specify the `-en` option if you want dbping to exit with a failed return code when a property value is unknown. See “Ping utility (dbping)” [*SQL Anywhere Database Administration Guide*, page 603].

- ◆ **Environment variables renamed** The following environment variables have been renamed for this release:

Previous name	New name
ASTMP	SATMP
ASDIR	SADIR
ASLOGDIR	SALOGDIR
ASLANG	SALANG
ASCHARSET	SACHARSET

- ◆ **Changes to PHP module file names** The naming convention for the PHP module files has been changed. In previous versions, the files were named *phpX_sqlanywhereY.dll*, where *X* was the PHP major version number and *Y* was the major SQL Anywhere version number. The PHP module files are now named *php-a.b.c_sqlanywhereY.dll*, where *a.b.c* is the full version number of the PHP source the file is built against and *Y* is the major SQL Anywhere version number. For example, *php-5.0.2_sqlanywhere10.dll*.
- ◆ **Specifying values for the PrefetchBuffer connection parameter** The PrefetchBuffer connection parameter now interprets values less than 16384 as kilobytes for backwards compatibility. Using kilobytes without the *k* suffix is deprecated. If the value of PrefetchBuffer is adjusted because it was out of the valid range or specified in kilobytes without the *k* suffix, the client log file shows the actual PrefetchBuffer value used. See “PrefetchBuffer connection parameter [PBUF]” [*SQL Anywhere Database Administration Guide*, page 227].
- ◆ **System-defined domains cannot be dropped** You can no longer drop system-defined domains, such as MONEY or UNIQUEIDENTIFIERSTR, from a database. See “DROP statement” [*SQL Anywhere Reference*, page 475].

- ◆ **Changes to database utilities** Following is a list of changes to the database utilities, as described:
 - **The Service utility (dbsvc) can grant the Login as a Service privilege** The Service utility (dbsvc) prompts you to grant the Login as a Service privilege if the -a option is used and you try to run a service under an account that does not have the Login as a Service privilege enabled. If the -y option is used, dbsvc attempts to grant the Login as a Service privilege without prompting you. See “[Service utility \(dbsvc\)](#)” [*SQL Anywhere Database Administration Guide*, page 616].
 - **The Unload utility (dbunload) -an option can be used against a remote server** Prior to this change you could only run dbunload -an against a server on the same computer. Now you can run dbunload -an against a server that is running on a different computer. See “[Unload utility \(dbunload\)](#)” [*SQL Anywhere Database Administration Guide*, page 644].
 - **The Server Enumeration utility (dblocate) host name or IP address formats** The host name or IP address can use any format, regardless of whether -n is specified. For example, if a server is running on myhost.mycompany.com, which has IP address of 1.2.3.4, to list only servers running on this computer from any computer with the mycompany.com domain, any of dblocate myhost, dblocate myhost.mycompany.com, or dblocate 1.2.3.4 can be used. In previous versions, only dblocate myhost.mycompany.com or dblocate -n 1.2.3.4 would have worked since the given hostname or IP address had to match the address string (excluding the port number) displayed by dblocate.
- ◆ **Default-related changes** The following changes have been made to defaults:
 - **Default TCP/IP port changed for personal database server** On Windows, the personal database server now listens for connections on port 127.0.0.1, rather than port 0.0.0.0. This change means that users running SQL Anywhere on Windows XP SP2 with the firewall enabled do not need to add dbeng10 to the exception list before it can be used. As a result of this change, trying to connect with `LINKS=tcip(HOST=hostname;DOBROADCAST=none)` will not work if *hostname* is the real host name or IP address of the computer. However, using a hostname of **localhost** or **127.0.0.1** will work.
 - **Default database page size changed to 4096** The default database page size for SQL Anywhere databases has been changed to 4096 bytes from 2048 bytes. This page size has been shown to improve performance in many environments. See “[CREATE DATABASE statement](#)” [*SQL Anywhere Reference*, page 349].

- **Default maximum cache size changes** The default, maximum cache size on Windows (non-AWE) has been increased. The default maximum cache size is now limited to the lesser of:
 - ◆ 90% of (*total_physical_memory* - 4 MB), but no less than 2 MB
 - ◆ (*available address space* - 512 MB)
- **Unix** The way the maximum cache size is calculated on Unix has changed. The default maximum cache size is now calculated as follows:
 - ◆ On 32-bit Unix platforms, it is the lesser of 90% of total physical memory or 1,834,880 KB.
On 64-bit Unix platforms, it is the lesser of 90% of total physical memory and 8,589,672,320 KB.

See “Limiting the memory used by the cache” [[SQL Anywhere SQL User's Guide, page 226](#)] and “-ch server option” [[SQL Anywhere Database Administration Guide, page 133](#)].

- **Default size when converting NULL constants to NUMERIC or string data types** When converting a NULL constant to the NUMERIC data type, or to string data types such as CHAR, and VARCHAR, the length is now set to 0, instead of 32767.
- **Default URI for openxml system procedure has changed** When using the openxml system procedure, if a namespace declaration is not specified, then by default the prefix mp is bound to the Uniform Resource Identifier (URI). In previous releases of the software, this URI was urn:ianywhere-com:asa-xpath-metaprop. The default URI has been renamed to urn:ianywhere-com:sa-xpath-metaprop. See “openxml system procedure” [[SQL Anywhere Reference, page 831](#)].
- ◆ **Changes to cache size percentage calculation for -c, -ch-, and -cl server options** When using P (percentage) with -c, -ch, or -cl, the system now calculates percentage against either the amount of physical system memory, or the amount of available address space, whichever is lower. This eliminates the risk of allocating more memory for the cache than is available for addressing. See “-c server option” [[SQL Anywhere Database Administration Guide, page 129](#)], “-ch server option” [[SQL Anywhere Database Administration Guide, page 133](#)], and “-cl server option” [[SQL Anywhere Database Administration Guide, page 134](#)].
- ◆ **SQL Anywhere OLE DB provider and data source name have changed** The SQL Anywhere OLE DB provider (previously, ASAProv), is now called SAOLEDB. The corresponding ODBC data source (previously ASA 9.0 Sample) is now called SQL Anywhere 10

Demo. See “OLE DB providers” [[SQL Anywhere Database Administration Guide](#), page 65].

- ◆ **Procedure_profiling server option renamed** The correct name of the server option that controls procedure profiling is now ProcedureProfiling. The previous form, Procedure_profiling, is still accepted, but will be unsupported in a future release. See “[sa_server_option system procedure](#)” [[SQL Anywhere Reference](#), page 907].
- ◆ **TCP/IP port number does not need to be specified for network database servers on HP-UX when the default port is not in use** In previous versions of the software, if you started a database server on HP-UX, you had to specify a port number using the ServerPort [PORT] protocol option if the default port (2638) was already in use or if you did not want to use the default port.
On HP-UX, the TCP/IP ServerPort protocol option is no longer required when multiple database servers are started on one machine. On Mac OS X, the TCP/IP ServerPort option must still be specified when starting a network server if a server is already running on the same computer. See “[ServerPort protocol option \[PORT\]](#)” [[SQL Anywhere Database Administration Guide](#), page 247].
- ◆ **Unload Database wizard behavior changes** You can no longer unload a database into an database version earlier than version 10. When you unload a version 9.0.2 or earlier database into a version 10 database, you cannot connect to database automatically once the rebuild completes.
- ◆ **Extract Database wizard behavior changes** You cannot extract version 9.0.2 and earlier databases. You must extract from a version 10 database.

Database option changes

- ◆ **Case sensitivity and database options** The SET OPTION statement and CONNECTION_PROPERTY function use case insensitive option names. However, in databases that use a Turkish collation or are case sensitive, option names referenced in queries should be written using the case specified in “[Database Options](#)” [[SQL Anywhere Database Administration Guide](#), page 351].

In either of these situations, executing a query on SYSOPTION or a query like the following may not match any rows if the option name is used with the wrong case

```
SELECT *
FROM sa_conn_properties()
WHERE propname = 'BLOCKING'
```

- ◆ **ansi_blanks option default setting changed** When a new database is created, the default value for the ansi_blanks database option is On. In previous versions of the software, the default value for this option was Off. For embedded SQL with ansi_blanks set to On and a blank padded database, when you supply a value of data type DT_STRING, you must set the sqllen field to the length of the buffer containing the value (at least the length of the value plus space for the terminating null character).

When a database is blank padded, the ansi_blanks option controls truncation warnings sent to the client if the expression being fetched is CHAR or NCHAR (not VARCHAR or NVARCHAR) and it is being fetched into a char or nchar (not VARCHAR or NVARCHAR) host variable.

See “ansi_blanks option [compatibility]” [*SQL Anywhere Database Administration Guide*, page 372].

- ◆ **ansi_integer_overflow option default setting changed** When a new database is created, the default value for the ansi_integer_overflow database option is On. In previous versions of the software, the default value for this option was Off. See “ansi_integer_overflow option [compatibility]” [*SQL Anywhere Database Administration Guide*, page 373].

- ◆ **date_format option no longer supports time-related information** The date_format option no longer supports the following values when specifying the format string:

- **hh** two digit hours
- **nn** two digit minutes
- **ss[.ss..]** seconds and parts of a second
- **aa** morning/afternoon indicator (A.M. or P.M., 12 hour clock)
- **aaa[a...]** morning/afternoon indicator (A.M. or P.M., 12 hour clock)
- **pp** afternoon indicator, if necessary (P.M., 12 hour clock)
- **ppp[p...]** afternoon indicator, if necessary (P.M., 12 hour clock)

- ◆ **login_mode database option** The value Mixed is deprecated for the login_mode database option. Specify Standard, Integrated to allow both standard and integrated logins. See “login_mode option [database]” [*SQL Anywhere Database Administration Guide*, page 408].

- ◆ **string_rtruncation option default setting changed** When a new database is created, the default value for the string_rtruncation database option is On. In previous versions of the software, the default value for this option was Off. See “string_rtruncation option [compatibility]” [*SQL Anywhere Database Administration Guide*, page 442].

- ◆ **temp_space_limit_check option default setting changed** The default setting for the temp_space_limit_check option has been changed to On. Now, by default, if a connection requests more than its quota of temporary file space, then the request fails and the error SQLSTATE_TEMP_SPACE_LIMIT is returned. See “temp_space_limit_check option [database]” [*SQL Anywhere Database Administration Guide*, page 444].
- ◆ **timestamp_format option no longer supports time-related information** The timestamp_format option no longer supports the use of French days and months. See “timestamp_format option [compatibility]” [*SQL Anywhere Database Administration Guide*, page 446].

Server option changes

- ◆ **-ec server option TLS syntax for strong encryption types has changed** The syntax for the strong encryption types in the -ec server switch and the ENCRYPTION connection parameter has changed. There are now only 3 types available: none, simple, and tls. Instead of specifying the key exchange algorithm to use as the type, you now specify tls as the encryption type, and use a new protocol option, tls_type, to specify the algorithm. See “-ec server option” [*SQL Anywhere Database Administration Guide*, page 141].
- ◆ **-os server option** In previous releases, the -os database server option renamed the log file to *current-file-name.old*. This is now the behavior of the -on option. The -os database server option now specifies a maximum size for the output log, at which point the log is renamed. Previously, using -os would result in two log files, but now it results in an unlimited number of log files. See “-os server option” [*SQL Anywhere Database Administration Guide*, page 161] and “-on server option” [*SQL Anywhere Database Administration Guide*, page 161].

Catalog changes

The catalog has undergone major changes in version 10.0.0. The most significant change is that system tables have been renamed to include an I at the beginning of their name. If you attempt to access the system tables, you will receive a permission denied error. Instead, the information contained in the system tables is made available through system views. There is one system view per system table, and, for backward compatibility, the system view names coincide with the table names from previous versions of SQL Anywhere. For example, in 9.0.2, there was a system table called SYS.SYSARTICLE. In version 10.0.0 that system table is now called

SYS.ISYSARTICLE. It is not accessible for querying; however, it has a corresponding system view, SYS.SYSARTICLE.

The catalog now also contains consolidated views. These are views which provide commonly needed joins from two or more tables or views. Most of the consolidated views were present as system views in previous releases.

Some system tables and views have been deprecated or removed from the catalog. In most cases, however, compatibility views are provided.

The following table provides a complete mapping of the catalog from Adaptive Server Anywhere 9.0.2 to SQL Anywhere 10.0.0. The first column, **9.0.2 system table/view**, shows names of the 9.0.2 system tables, followed by a forward slash ('/'), and then the name the 9.0.2 associated view(s). The middle column, **10.0.0 system table**, contains the 10.0.0 table name. The final column, **10.0.0 system view**, contains the associated 10.0.0 view name(s), as well as compatibility notes.

Note
A dash (-) in any of the columns indicates that there is no equivalent object. For example, a new table in the catalog for the 10.0.0 release results in a dash for the table in the 9.0.2 column.

9.0.2 system table/view	10.0.0 system table	10.0.0 system view
DUMMY / -	DUMMY	-
RowGenerator / -	RowGenerator	-
SYSARTICLE / SYSARTICLES	ISYSARTICLE	“SYSARTICLE system view” [SQL Anywhere Reference, page 741] For pre-10.0.0 compatibility: “SYSARTICLES consolidated view” [SQL Anywhere Reference, page 799] .
SYSARTICLECOL / SYSARTICLECOL	ISYSARTICLECOL	“SYSARTICLECOL system view” [SQL Anywhere Reference, page 741] For pre-10.0.0 compatibility: “SYSARTICLECOLS consolidated view” [SQL Anywhere Reference, page 799] .
SYSATTRIBUTE / -	ISYSATTRIBUTE	-
SYSATTRIBUTENAME / -	-	-

9.0.2 system table/view	10.0.0 system table	10.0.0 system view
SYSCAPABILITY / SYSCAPABILITIES	ISYSCAPABILITY	“SYSCAPABILITY system view” [<i>SQL Anywhere Reference</i> , page 742] “SYSCAPABILITIES consolidated view” [<i>SQL Anywhere Reference</i> , page 800]
SYSCAPABILITYNAME / - - / SYSCATALOG	ISYSCAPABILITY- NAME	“SYSCAPABILITYNAME system view” [<i>SQL Anywhere Reference</i> , page 743] “SYSCATALOG consolidated view” [<i>SQL Anywhere Reference</i> , page 800]
SYSCHECK / -	ISYSCHECK	“SYSCHECK system view” [<i>SQL Anywhere Reference</i> , page 743]
- / SYSCOLAUTH	-	“SYSCOLAUTH consolidated view” [<i>SQL Anywhere Reference</i> , page 800]
SYSCOLLATION / -	-	“SYSCOLLATION compatibility view (deprecated)” [<i>SQL Anywhere Reference</i> , page 817]
SYSCOLLATIONMAP- PINGS / -	-	“SYSCOLLATIONMAPPINGS compatibility view (deprecated)” [<i>SQL Anywhere Reference</i> , page 817]
SYSCOLPERM / -	ISYSCOLPERM	“SYSCOLPERM system view” [<i>SQL Anywhere Reference</i> , page 743]
SYSCOLSTAT / SYSCOLSTATS	ISYSCOLSTAT	“SYSCOLSTAT system view” [<i>SQL Anywhere Reference</i> , page 744] and “SYSCOLSTATS consolidated view” [<i>SQL Anywhere Reference</i> , page 801]
SYSCOLUMN / SYSCOLUMNS	ISYSTABCOL	“SYSTABCOL system view” [<i>SQL Anywhere Reference</i> , page 786] and “SYSCOLUMNS consolidated view” [<i>SQL Anywhere Reference</i> , page 802] For pre-10.0.0 compatibility: “SYSCOLUMN compatibility view (deprecated)” [<i>SQL Anywhere Reference</i> , page 818]
SYSCONSTRAINT / -	ISYSCONSTRAINT	“SYSCONSTRAINT system view” [<i>SQL Anywhere Reference</i> , page 746]
- / -	ISYSDEPENDENCY	“SYSDEPENDENCY system view” [<i>SQL Anywhere Reference</i> , page 747]

9.0.2 system table/view	10.0.0 system table	10.0.0 system view
SYSDOMAIN / -	ISYSDOMAIN	“SYSDOMAIN system view” [<i>SQL Anywhere Reference</i> , page 747]
SYSEVENT / -	ISYSEVENT	“SYSEVENT system view” [<i>SQL Anywhere Reference</i> , page 748]
SYSEVENTTYPE / -	ISYSEVENTTYPE	“SYSEVENTTYPE system view” [<i>SQL Anywhere Reference</i> , page 749]
SYSEXTENT / -	-	-
SYSEXTERNLOGINS / -	ISYSEXTERNLOGIN	“SYSEXTERNLOGIN system view” [<i>SQL Anywhere Reference</i> , page 750]
SYSFILE / -	ISYSFILE	“SYSFILE system view” [<i>SQL Anywhere Reference</i> , page 751]
SYSFKCOL / -	ISYSIDXCOL	“SYSIDXCOL system view” [<i>SQL Anywhere Reference</i> , page 756] For pre-10.0.0 compatibility: “SYSFKCOL compatibility view (deprecated)” [<i>SQL Anywhere Reference</i> , page 818]
SYSFOREIGNKEY / SYSFOREIGNKEYS	ISYSFKEY	“SYSFKEY system view” [<i>SQL Anywhere Reference</i> , page 751] and “SYSFOREIGNKEYS consolidated view” [<i>SQL Anywhere Reference</i> , page 802]. For pre-10.0.0 compatibility: “SYSFOREIGNKEY compatibility view (deprecated)” [<i>SQL Anywhere Reference</i> , page 819]
- / SYSGROUPS	ISYSGROUP	“SYSGROUP system view” [<i>SQL Anywhere Reference</i> , page 753] and “SYSGROUPS consolidated view” [<i>SQL Anywhere Reference</i> , page 803]
SYSHISTORY / -	ISYSHISTORY	“SYSHISTORY system view” [<i>SQL Anywhere Reference</i> , page 753]
SYSINDEX / SYSINDEXES	ISYSIDX	“SYSIDX system view” [<i>SQL Anywhere Reference</i> , page 755] and “SYSINDEXES consolidated view” [<i>SQL Anywhere Reference</i> , page 803] For pre-10.0.0 compatibility: “SYSINDEX compatibility view (deprecated)” [<i>SQL Anywhere Reference</i> , page 820]

9.0.2 system table/view	10.0.0 system table	10.0.0 system view
SYSINFO / -	-	“SYSINFO compatibility view (deprecated)” [<i>SQL Anywhere Reference</i> , page 821]
SYSIXCOL / -	ISYSIDXCOL	“SYSIDXCOL system view” [<i>SQL Anywhere Reference</i> , page 756] For pre-10.0.0 compatibility: “SYSIXCOL compatibility view (deprecated)” [<i>SQL Anywhere Reference</i> , page 821]
SYSJAR / -	ISYSJAR	“SYSJAR system view” [<i>SQL Anywhere Reference</i> , page 757]
SYSJARCOMPONENT / -	ISYSJARCOMPONENT	“SYSJARCOMPONENT system view” [<i>SQL Anywhere Reference</i> , page 758]
SYSJAVACLASS / -	ISYSJAVACLASS	“SYSJAVACLASS system view” [<i>SQL Anywhere Reference</i> , page 759]
SYSLOGIN / -	ISYSLOGINMAP	“SYSLOGINMAP system view” [<i>SQL Anywhere Reference</i> , page 760]
SYSOPTBLOCK / -	-	system use only
- / -	ISYSMVOPTION	“SYSMVOPTION system view” [<i>SQL Anywhere Reference</i> , page 761]
- / -	ISYSMVOPTIONNAME	“SYSMVOPTIONNAME system view” [<i>SQL Anywhere Reference</i> , page 762]
- / -	ISYSOBJECT	“SYSOBJECT system view” [<i>SQL Anywhere Reference</i> , page 762]
SYSOPTION / SYSOPTIONS	ISYSOPTION	“SYSOPTION system view” [<i>SQL Anywhere Reference</i> , page 764] and “SYSOPTIONS consolidated view” [<i>SQL Anywhere Reference</i> , page 804]
SYSOPTJOINSTRATEGY / SYSOPTJOINSTRATEGIES	-	system use only
SYSOPTORDER / SYSOPTORDERS	-	system use only
SYSOPTQUANTIFIER / -	-	system use only
SYSOPTREQUEST / -	-	system use only
SYSOPTREWRITE / -	-	system use only

9.0.2 system table/view	10.0.0 system table	10.0.0 system view
SYSOPTSTAT / -	ISYSOPTSTAT	“SYSOPTSTAT system view” [<i>SQL Anywhere Reference</i> , page 764]
-	ISYSPHYSIDX	“SYSPHYSIDX system view” [<i>SQL Anywhere Reference</i> , page 765]
- / SYSPROCAUTH	-	“SYSPROCAUTH consolidated view” [<i>SQL Anywhere Reference</i> , page 805]
SYSPROCEDURE / SYSPROCEDURES	ISYSPROCEDURE	“SYSPROCEDURE system view” [<i>SQL Anywhere Reference</i> , page 766] The SYSPROCEDURES view has been renamed to SYSPROCS. See “SYSPROCS consolidated view” [<i>SQL Anywhere Reference</i> , page 805].
SYSROPCPARM / SYSPROCPARMS	ISYSROPCPARM	“SYSROPCPARM system view” [<i>SQL Anywhere Reference</i> , page 767] and “SYSPROCPARMS consolidated view” [<i>SQL Anywhere Reference</i> , page 805]
SYSROPCPERM / -	ISYSROPCPERM	“SYSROPCPERM system view” [<i>SQL Anywhere Reference</i> , page 769]
-	ISYSROXYTAB	“SYSROXYTAB system view” [<i>SQL Anywhere Reference</i> , page 770]
SYSPUBLICATION / SYSPUBLICATIONS	ISYSPUBLICATION	“SYSPUBLICATION system view” [<i>SQL Anywhere Reference</i> , page 770] and “SYSPUBLICATIONS consolidated view” [<i>SQL Anywhere Reference</i> , page 806]
- / -	ISYSREMARK	“SYSREMARK system view” [<i>SQL Anywhere Reference</i> , page 772]
SYSREMOTEOPTION / SYSREMOTEOPTIONS, SYSREMOTEOPTION2	ISYSREMOTEOPTION	“SYSREMOTEOPTION system view” [<i>SQL Anywhere Reference</i> , page 772], “SYSREMOTEOPTION2 consolidated view” [<i>SQL Anywhere Reference</i> , page 807], and “SYSREMOTEOPTIONS consolidated view” [<i>SQL Anywhere Reference</i> , page 807]
SYSREMOTEOPTION- TYPE / -	ISYSREMOTEOPTION- TYPE	“SYSREMOTEOPTIONTYPE system view” [<i>SQL Anywhere Reference</i> , page 773]

9.0.2 system table/view	10.0.0 system table	10.0.0 system view
SYSREMOTETYPE / SYSREMOTETYPES	ISYSREMOTETYPE	“SYSREMOTETYPE system view” [<i>SQL Anywhere Reference</i> , page 773] and “SYSREMOTETYPES consolidated view” [<i>SQL Anywhere Reference</i> , page 808]
SYSREMOTEUSER / SYSREMOTEUSERS	ISYSREMOTEUSER	“SYSREMOTEUSER system view” [<i>SQL Anywhere Reference</i> , page 774] and “SYSREMOTEUSERS consolidated view” [<i>SQL Anywhere Reference</i> , page 808]
SYSSCHEDULE / -	ISYSSCHEDULE	“SYSSCHEDULE system view” [<i>SQL Anywhere Reference</i> , page 776]
SYSSERVERS / -	ISYSSERVER	“SYSSERVER system view” [<i>SQL Anywhere Reference</i> , page 777]
- / -	ISYSSOURCE	“SYSSOURCE system view” [<i>SQL Anywhere Reference</i> , page 778]
SYSSQLSERVERTYPE / -	ISYSSQLSERVERTYPE	“SYSSQLSERVERTYPE system view” [<i>SQL Anywhere Reference</i> , page 779]
SYSSUBSCRIPTION / SYSSUBSCRIPTIONS	ISYSSUBSCRIPTION	“SYSSUBSCRIPTION system view” [<i>SQL Anywhere Reference</i> , page 779] and “SYSSUBSCRIPTIONS consolidated view” [<i>SQL Anywhere Reference</i> , page 809]
SYSSYNC / SYSSYNCS, SYSSYNC2	ISYSSYNC	“SYSSYNC system view” [<i>SQL Anywhere Reference</i> , page 780], “SYSSYNC2 consolidated view” [<i>SQL Anywhere Reference</i> , page 809], and “SYSSYNC2 consolidated view” [<i>SQL Anywhere Reference</i> , page 809]
-	ISYSSYNCSRIPT	“SYSSYNCSRIPT system view” [<i>SQL Anywhere Reference</i> , page 782] and “SYSSYNCSRIPTS consolidated view” [<i>SQL Anywhere Reference</i> , page 811]
- / SYSSYNCSUBSCRIPTIONS	-	“SYSSYNCSUBSCRIPTIONS consolidated view” [<i>SQL Anywhere Reference</i> , page 811]
- / SYSSYNCSUSERS	-	“SYSSYNCSUSERS consolidated view” [<i>SQL Anywhere Reference</i> , page 812]
- / SYSTABAUTH	-	“SYSTABAUTH consolidated view” [<i>SQL Anywhere Reference</i> , page 812]

9.0.2 system table/view	10.0.0 system table	10.0.0 system view
SYSTABLE / -	ISYSTAB	“SYSTAB system view” [<i>SQL Anywhere Reference</i> , page 783] For pre-10.0.0 compatibility: “SYSTABLE compatibility view (deprecated)” [<i>SQL Anywhere Reference</i> , page 822]
-	ISYSTABCOL	“SYSTABCOL system view” [<i>SQL Anywhere Reference</i> , page 786]
SYSTABLEPERM / -	ISYSTABLEPERM	“SYSTABLEPERM system view” [<i>SQL Anywhere Reference</i> , page 788]
SYSTRIGGER / SYSTRIGGERS	ISYSTRIGGER	“SYSTRIGGER system view” [<i>SQL Anywhere Reference</i> , page 790] and “SYSTRIGGERS consolidated view” [<i>SQL Anywhere Reference</i> , page 813]
SYSTYPEMAP / -	ISYSTYPEMAP	“SYSTYPEMAP system view” [<i>SQL Anywhere Reference</i> , page 792]
-	ISYSUSER	“SYSUSER system view” [<i>SQL Anywhere Reference</i> , page 793]
- / SYSUSERAUTH	ISYSUSERAUTHORITY	“SYSUSERAUTHORITY system view” [<i>SQL Anywhere Reference</i> , page 793] and “SYSUSERAUTH consolidated view” [<i>SQL Anywhere Reference</i> , page 814]
- / SYSUSERLIST		“SYSUSERLIST consolidated view” [<i>SQL Anywhere Reference</i> , page 814]
SYSUSERMESSAGES / -	ISYSUSERMESSAGE	“SYSUSERMESSAGE system view” [<i>SQL Anywhere Reference</i> , page 794]
- / SYSUSEROPTIONS	-	“SYSUSEROPTIONS consolidated view” [<i>SQL Anywhere Reference</i> , page 815]

9.0.2 system table/view	10.0.0 system table	10.0.0 system view
SYSUSERPERM / SYSUSERPERMS	-	Data now located in the ISYSUSER and ISYSUSERAUTHORITY system tables. See: “SYSUSER system view” [SQL Anywhere Reference, page 793] and “SYSUSERAUTHORITY system view” [SQL Anywhere Reference, page 793] For pre-10.0.0 compatibility: “SYSUSERPERM compatibility view (deprecated)” [SQL Anywhere Reference, page 824] and “SYSUSERPERMS compatibility view (deprecated)” [SQL Anywhere Reference, page 824]
SYSUSERTYPE / -	ISYSUSERTYPE	“SYSUSERTYPE system view” [SQL Anywhere Reference, page 794]
- / SYSVIEWS	ISYSVIEW	“SYSVIEW system view” [SQL Anywhere Reference, page 795] and “SYSVIEWS consolidated view” [SQL Anywhere Reference, page 815]
SYSWEBSERVICE / -	ISYSWEBSERVICE	“SYSWEBSERVICE system view” [SQL Anywhere Reference, page 797]

Summary of new views

System view name	Link to more information
SYSDEPENDENCY	Each row in the SYSDEPENDENCY system view describes a dependency between two database objects. See “SYSDEPENDENCY system view” [SQL Anywhere Reference, page 747]
SYSFKEY	Each row in the SYSFKEY system view describes a foreign key constraint in the system. See “SYSFKEY system view” [SQL Anywhere Reference, page 751].
SYSIDX	Each row in the SYSIDX system table defines a logical index in the database. See “SYSIDX system view” [SQL Anywhere Reference, page 755].
SYSIDXCOL	Each row in the SYSIDXCOL system view describes one column of an index described in the SYSIDX system view. See “SYSIDXCOL system view” [SQL Anywhere Reference, page 756].
SYSLOGINMAP	The SYSLOGINMAP system view contains all the user names that can be used to connect to the database using either an integrated login, or Kerberos login. See “SYSLOGINMAP system view” [SQL Anywhere Reference, page 760].

System view name	Link to more information
SYSMVOPTION	Each row in the SYSMVOPTION system view describes the setting of one option value for a materialized view. See “SYSMVOPTION system view” [SQL Anywhere Reference, page 761] .
SYSMVOPTION-NAME	Each row in the SYSMVOPTIONNAME system view contains the name of an option defined in the SYSMVOPTION system view. See “SYSMVOPTION-NAME system view” [SQL Anywhere Reference, page 762] .
SYSOBJECT	Each row in the SYSOBJECT system view describes an object. Examples of database objects include tables, views, columns, indexes, and procedures. See “SYSOBJECT system view” [SQL Anywhere Reference, page 762] .
SYSPHYSIDX	Each row in the SYSPHYSIDX system view defines a physical index in the database. See “SYSPHYSIDX system view” [SQL Anywhere Reference, page 765] .
SYSPROCS	The SYSPROCS system view replaces the former SYSPROCEDURES view. See “SYSPROCS consolidated view” [SQL Anywhere Reference, page 805] .
SYSPROXYTAB	Each row of the SYSPROXYTAB system view describes the remote parameters of one proxy table. See “SYSPROXYTAB system view” [SQL Anywhere Reference, page 770] .
SYSREMARK	Each row in the SYSREMARK system view describes a remark (or comment) for an object. See “SYSREMARK system view” [SQL Anywhere Reference, page 772] .
SYSSOURCE	Each row in the SYSSOURCE system view contains the source for an object listed in the ISYSOBJECT system table. See “SYSSOURCE system view” [SQL Anywhere Reference, page 778] .
SYSSYNCSCRIPT	Each row in the SYSSYNCSCRIPT system view identifies a stored procedure for MobiLink scripted upload. See “SYSSYNCSCRIPT system view” [SQL Anywhere Reference, page 782] .
SYSTABCOL	The SYSTABCOL system view contains one row for each column of each table and view in the database. See “SYSTABCOL system view” [SQL Anywhere Reference, page 786] .
SYSUSER	Each row in the SYSUSER system view describes a user in the system. See “SYSUSER system view” [SQL Anywhere Reference, page 793] .
SYSUSERAUTHORITY	Each row of SYSUSERAUTHORITY system view describes the authority granted to one user ID. See “SYSUSERAUTHORITY system view” [SQL Anywhere Reference, page 793] .

Summary of deprecated tables or views Following is a list of catalog objects that are deprecated. In most cases, the object was a table in previous versions but is now a compatibility view. Referencing these objects does not result in an error, however, for future compatibility, you are encouraged to change your applications to point to the suggested object(s) instead.

Deprecated table or view	Transition information
SYSCOLLATION system table	Collation mapping information is now stored as database properties. See “ SYSCOLLATION compatibility view (deprecated) ” [<i>SQL Anywhere Reference</i> , page 817].
SYSCOLLATIONMAPPINGS system table	Collation mapping information is now stored as database properties. See “ SYSCOLLATIONMAPPINGS compatibility view (deprecated) ” [<i>SQL Anywhere Reference</i> , page 817].
SYSCOLUMN system table	Use the SYSTABCOL system view instead. See “ SYSTABCOL system view ” [<i>SQL Anywhere Reference</i> , page 786] and “ SYSCOLUMN compatibility view (deprecated) ” [<i>SQL Anywhere Reference</i> , page 818].
SYSFKCOL system table	Use the SYSFKEY system view instead. See “ SYSFKEY system view ” [<i>SQL Anywhere Reference</i> , page 751] and “ SYSFKCOL compatibility view (deprecated) ” [<i>SQL Anywhere Reference</i> , page 818].
SYSFOREIGNKEY system table	Use the SYSFKEY system view instead. See “ SYSFKEY system view ” [<i>SQL Anywhere Reference</i> , page 751] and “ SYSFOREIGNKEY compatibility view (deprecated) ” [<i>SQL Anywhere Reference</i> , page 819].
SYSINDEX system table	Use the SYSIDX system view instead. See “ SYSIDX system view ” [<i>SQL Anywhere Reference</i> , page 755] and “ SYSINDEX compatibility view (deprecated) ” [<i>SQL Anywhere Reference</i> , page 820].
SYSIXCOL system table	Use the SYSIDXCOL system view instead. See “ SYSIDXCOL system view ” [<i>SQL Anywhere Reference</i> , page 756] and “ SYSIXCOL compatibility view (deprecated) ” [<i>SQL Anywhere Reference</i> , page 821].
SYSTABLE system table	Use the SYSTAB system view instead. See “ SYSTAB system view ” [<i>SQL Anywhere Reference</i> , page 783] and “ SYSTABLE compatibility view (deprecated) ” [<i>SQL Anywhere Reference</i> , page 822].

Deprecated table or view	Transition information
SYSUSERPERM system table	Use the SYSUSERAUTHORITY system view instead. See “SYSUSERAUTHORITY system view” [SQL Anywhere Reference, page 793] and “SYSUSERPERM compatibility view (deprecated)” [SQL Anywhere Reference, page 824] .
SYSUSERPERMS system view	Use the SYSUSERAUTHORITY system view instead. See “SYSUSERAUTHORITY system view” [SQL Anywhere Reference, page 793] and “SYSUSERPERMS compatibility view (deprecated)” [SQL Anywhere Reference, page 824] .

Summary of removed or renamed tables or views Following is a list of catalog objects that are no longer present in the catalog. Referencing these objects results in an error.

Removed table or view	Transition information
SYSATTRIBUTE system table	Use the SYSIDX and SYSPHYSIDX system views instead. Information about percent free and clustered index is now maintained in the ISYSIDX system table. Information about key values, key distance, leaf pages, and depth is now stored in the ISYSPHYSIDX system table. See “SYSIDX system view” [SQL Anywhere Reference, page 755] and “SYSPHYSIDX system view” [SQL Anywhere Reference, page 765] .
SYSATTRIBUTENAME system table	Use the SYSIDX and SYSPHYSIDX system views instead. See “SYSIDX system view” [SQL Anywhere Reference, page 755] and “SYSPHYSIDX system view” [SQL Anywhere Reference, page 765] .
SYSEXTENT system table	The SYSEXTENT table is no longer available in the catalog in SQL Anywhere version 10.0.0 and higher. This table was previously unused.
SYSEXTERNLOGINS	Renamed to SYSEXTERNLOGIN. See “SYSEXTERNLOGIN system view” [SQL Anywhere Reference, page 750] .
SYSLOGIN system table	The SYSLOGIN table has been replaced by the SYSLOGINMAP system view, with some changes. See “SYSLOGINMAP system view” [SQL Anywhere Reference, page 760] .
SYSOPTBLOCK	This table was for internal use only.
SYSOPTJOINSTRATEGY	This table was for internal use only.
SYSOPTJOINSTRATEGIES	This view was for internal use only.
SYSOPTORDER	This table was for internal use only.
SYSOPTORDERS	This view was for internal use only.
SYSOPTQUANTIFIER	This table was for internal use only.

Removed table or view	Transition information
SYSOPTREQUEST	This table was for internal use only.
SYSOPTREWRITE	This table was for internal use only.
SYS PROCEDURES view	Use the SYSPROCS consolidated view instead. See “ SYSPROCS consolidated view ” [SQL Anywhere Reference , page 805].
SYSSERVERS	Renamed to SYSSERVER. See “ SYSSERVER system view ” [SQL Anywhere Reference , page 777].
SYSUSERMESSAGES	Renamed to SYSUSERMESSAGE. See “ SYSUSERMESSAGE system view ” [SQL Anywhere Reference , page 794].

Change to columns in system tables and system views

There have been numerous changes to columns in system tables and views. With the exception of the information below, all of the changes consist of adding new columns, or removing unused columns, neither of which impact your applications.

- ◆ **SYSCONSTRAINT** The previous SYSCONSTRAINT system table has been replaced by a new system table, ISYSCONSTRAINT, with a corresponding SYSCONSTRAINT system view. References to SYSCONSTRAINT will go to the new system view, which is significantly different in this release. To see the contents of the SYSCONSTRAINT system view, see “[SYSCONSTRAINT system view](#)” [[SQL Anywhere Reference](#), page 746].

SQL statements

- ◆ **Restrictions on key joins for derived tables** Key joins are not allowed for derived tables containing TOP N, START AT, FIRST, ORDER BY, window functions, FOR XML, or recursive tables. See “[Key joins of views and derived tables](#)” [[SQL Anywhere SQL User's Guide](#), page 373].
- ◆ **ALTER statements** All ALTER statements now use ALTER as a subclause, instead of MODIFY. If your applications use a MODIFY subclause, you should change them to use the ALTER subclause instead. The MODIFY syntax is still supported but deprecated. This impacts the following statements:
 - ◆ “[ALTER DATABASE statement](#)” [[SQL Anywhere Reference](#), page 271]
 - ◆ “[ALTER EVENT statement](#)” [[SQL Anywhere Reference](#), page 278]
 - ◆ “[ALTER PUBLICATION statement \[MobiLink\]](#)” [[SQL Anywhere Reference](#), page 288]

- ◆ “[ALTER SYNCHRONIZATION SUBSCRIPTION statement \[MobiLink\]](#)” [*SQL Anywhere Reference*, page 299]
- ◆ “[ALTER SYNCHRONIZATION USER statement \[MobiLink\]](#)” [*SQL Anywhere Reference*, page 301]
- ◆ “[ALTER TABLE statement](#)” [*SQL Anywhere Reference*, page 303]

- ◆ **BACKUP statement** In previous releases, you could specify the DBFILE ONLY clause with either the TRANSACTION LOG RENAME or TRANSACTION LOG TRUNCATE clause. Specifying DBFILE ONLY with either of these TRANSACTION LOG clauses now results in an error because these are two exclusive types of backup. See “[BACKUP statement](#)” [*SQL Anywhere Reference*, page 318].

- ◆ **COMMENT statement** The syntax COMMENT ON LOGIN is no longer supported. Use the syntax COMMENT ON INTEGRATED LOGIN instead. See “[COMMENT statement](#)” [*SQL Anywhere Reference*, page 340].

- ◆ **VALIDATE statements** All validation activities, such as executing VALIDATE statements, or running the Validation utility (dbvalid), now require VALIDATE authority; REMOTE DBA permission is no longer accepted for performing validation activities.

The VALIDATE TABLE and VALIDATE MATERIALIZED VIEW statements now check for orphaned BLOBs. See “[VALIDATE statement](#)” [*SQL Anywhere Reference*, page 687].

The syntax for VALIDATE INDEX has changed to be consistent with the ALTER INDEX statement syntax. The old syntax is still supported, although deprecated. If your applications currently use the VALIDATE INDEX statement, you should change to the new syntax. See “[VALIDATE INDEX statement](#)” [*SQL Anywhere Reference*, page 690].

- ◆ **ALTER SERVER and CREATE SERVER statements** The ASAJDBC and ASAODBC server classes have been renamed to SAJDBC and SAODBC, respectively. See “[ALTER SERVER statement](#)” [*SQL Anywhere Reference*, page 291] and “[CREATE SERVER statement](#)” [*SQL Anywhere Reference*, page 411].

Deprecated and discontinued features

- ◆ **Transact-SQL outer joins deprecated** Transact-SQL outer joins have been deprecated in this release, and will not be supported in future versions of SQL Anywhere. The new connection option `tsql_outer_joins` enables or disables the ability to use the TSQL outer joins operators `*=` and `=*` in DML statements and views. This option is set to Off by default.

- ◆ **Unsupported properties** The NumProcessorsAvail and NumProcessorsMax server properties are no longer supported. You can use the NumLogicalProcessors, NumLogicalProcessorsUsed, NumPhysicalProcessors, and NumPhysicalProcessorsUsed server properties instead.
- ◆ **STRIP ON clause of LOAD TABLE is deprecated** While the stripping of leading and trailing blanks has been enhanced in SQL Anywhere 10.0.0 to allow you to fine tune the stripping behavior, STRIP ON is deprecated. To continue stripping trailing blanks *only* (default behavior in previous releases when STRIP ON was specified), use STRIP RTRIM instead. See [“LOAD TABLE statement” \[SQL Anywhere Reference, page 558\]](#).
- ◆ **UTF8 collation is deprecated** The UTF8 collation is deprecated. Use the UTF8BIN collation instead. See [“Supplied and recommended collations” \[SQL Anywhere Database Administration Guide, page 301\]](#).
- ◆ **jConnect 4.5 no longer supported** jConnect version 4.5 is no longer supported. Applications that previously connected using jConnect 4.5 must be changed to use jConnect 5.5 or 6.0 instead. See [“Using the jConnect JDBC driver” \[SQL Anywhere Programming Guide, page 339\]](#).
- ◆ **SQLLOCALE environment variable no longer supported** The SQLLOCALE environment variable is no longer supported. It has been replaced by the SALANG and SACHARSET environment variables. See [“SALANG environment variable” \[SQL Anywhere Database Administration Guide, page 268\]](#) and [“SACHARSET environment variable” \[SQL Anywhere Database Administration Guide, page 267\]](#).
- ◆ **Named Pipes no longer supported** The Named Pipes protocol is no longer supported. Applications that previously used Named Pipes must be changed to use shared memory instead. See [“Selecting communications protocols” \[SQL Anywhere Database Administration Guide, page 18\]](#).
- ◆ **Data Source utility (dbdsn) -o option deprecated** The -o option for the Data Source utility has been deprecated. If you want to write output messages to a file, you can specify the LogFile connection parameter in the connection string. See [“LogFile connection parameter \[LOG\]” \[SQL Anywhere Database Administration Guide, page 225\]](#).
- ◆ **Creation of custom collations not supported** Creation of custom collations is no longer supported. The Create Custom Collation wizard, the Collation utility (dbcollat), the DBCollate function, and the a_db_collation structure, are no longer supported. See [“Choosing collations” \[SQL Anywhere Database Administration Guide, page 300\]](#).

- ◆ **Database server -d option not supported** The -d database server option, used on NetWare to force the use of POSIX I/O rather than DFS (Direct File System) I/O is no longer supported.
- ◆ **Database server -y option not supported** The -y database server option, used on Windows 95/98/Me to run the database server as a Windows service is no longer supported because these operating systems are no longer supported. To run the database server as a service on any of the supported platforms, use the dbdsn utility. See [“Data Source utility \(dbdsn\)”](#) [*SQL Anywhere Database Administration Guide*, page 552].
- ◆ **-sc option not supported** SQL Anywhere 7.0 was awarded a TCSEC (Trusted Computer System Evaluation Criteria) C2 security rating from the U.S. Government. The -sc server option allowed you to run the current version of SQL Anywhere in a manner equivalent to the C2-certified environment. Support for the -sc option, as well as the C2 server property, has been removed in version 10.0.0.
- ◆ **max_work_table_hash_size database option not supported** The max_work_table_hash_size option is no longer supported. The query optimizer allocates hash sizes for the internal temporary tables based on the data distribution within the table.
- ◆ **max_hash_size database option not supported** The max_hash_size option is no longer supported.
- ◆ **Compressed databases and write files not supported** As a result, the following features are no longer available:
 - **File extensions** The following file extensions are no longer supported:
 - ◆ the .wrt extension used to identify write files
 - ◆ the .cdb extension used to identify compressed database files
 - **Database server behavior on NetWare** The database server no longer looks for database files with the .wrt extension when a database file is specified without an extension. See [“The SQL Anywhere database server”](#) [*SQL Anywhere Database Administration Guide*, page 120].
 - **Deploying databases on read-only media** You can no longer supply a write file to record changes to a database supplied on read-only media, such as a CD-ROM. However, you can still deploy databases on read-only media if they are run in read-only mode. See [“Deploying databases on read-only media”](#) [*SQL Anywhere Programming Guide*, page 693] and [“-r server option”](#) [*SQL Anywhere Database Administration Guide*, page 166].

- **Database utilities** The following utilities and wizards are no longer supported:
 - ◆ Compress Database wizard
 - ◆ Create Write File wizard
 - ◆ Uncompress Database wizard
 - ◆ Uncompression utility (dbexpand)
 - ◆ Compression utility (dbshrink)
 - ◆ Write File utility (dbwrite)
- **SQL statements** The following SQL statements are no longer supported:
 - ◆ ALTER WRITEFILE
 - ◆ CREATE WRITEFILE
 - ◆ CREATE COMPRESSED DATABASE
 - ◆ CREATE EXPANDED DATABASE
- **DBTools structures** The following structures or members of structures are no longer supported:
 - ◆ **a_backup_db structure** This structure holds the information needed to carry out backup tasks using the DBTools library. The backup_writefile member now appears as _unused.
 - ◆ **a_compress_db structure**
 - ◆ **a_compress_stats structure** This structure holds the information needed to carry out database compression tasks using the DBTools library.
 - ◆ **a_db_info structure** This structure holds the information needed to return dbinfo information using the DBTools library. The wrtbufsize member now appears as _unused1, the wrtnamebuffer member now appears as _unused2, and the compressed member now appears as _unused3.
 - ◆ **an_expand_db structure** This structure holds information needed for database expansion using the DBTools library.
 - ◆ **a_stats_line structure** This structure holds information needed for database compression and expansion using the DBTools library.
 - ◆ **a_writefile structure** This structure holds information needed for database write file management using the DBTools library.
- **DBTools functions** The following functions are no longer supported:
 - ◆ DBChangeWriteFile
 - ◆ DBCompress
 - ◆ DBCreateWriteFile
 - ◆ DBExpand
 - ◆ DBStatusWriteFile

- **Database properties** The following database properties are no longer supported:
 - ◆ Compression
 - ◆ FileSize *writefile*
 - ◆ FreePages *writefile*
- **DB_BACKUP_WRITEFILE** This embedded SQL function is no longer supported.
- ◆ **Support for unused ASE-compatibility views and procedures removed** Support for the following unused Adaptive Server Enterprise views in the SQL Anywhere database has been removed:

SYSALTERNATES	SYSLOGINROLES
SYSAUDITOPTIONS	SYSLOGS
SYSAUDITS	SYSMESSAGES
SYSCHARSETS	SYSPROCEDURES
SYSCONFIGURES	SYSPROCESSES
SYSCONSTRAINTS	SYSPROTECTS
SYSCURCONFIGS	SYSREFERENCES
SYSDATABASES	SYSREMOTEOLOGINS
SYSDEPENDS	SYSROLES
SYSDEVICES	SYSSEGMENTS
SYSENGINES	SYSSEVERERS
SYSKEYS	SYSSEVRROLES
SYSLANGUAGES	SYSTHRESHOLDS
SYSLOCKS	SYSUSAGES

Support for the following unused Adaptive Server Enterprise procedures in the SQL Anywhere database has been removed:

sp_addalias	sp_helpindex
sp_addauditrecord	sp_helpjoins
sp_addllanguage	sp_helpkey

sp_addremotelogin	sp_helplanguage
sp_addsegment	sp_helplog
sp_addserver	sp_helpremotelogin
sp_addthreshold	sp_helpprotect
sp_adddumpdevice	sp_helpsegment
sp_auditdatabase	sp_helpserver
sp_auditlogin	sp_helpsort
sp_auditobject	sp_helpthreshold
sp_auditooption	sp_helpuser
sp_auditsproc	sp_indsuspect
sp_bindefault	sp_lock
sp_bindmsg	sp_locklogin
sp_bindrule	sp_logdevice
sp_changedbowner	sp_modifylogin
sp_checknames	sp_modifythreshold
sp_checkreswords	sp_monitor
sp_clearstats	sp_placeobject
sp_commonkey	sp_primarykey
sp_configure	sp_procxmode
sp_cursorinfo	sp_recompile
sp_dboption	sp_remap
sp_dbremap	sp_remoteoption
sp_depends	sp_rename
sp_diskdefault	sp_renamedb
sp_displaylogin	sp_reportstats
sp_dropalias	sp_role
sp_dropdevice	sp_serveroption
sp_dropkey	sp_setlangalias

sp_droplanguage	sp_spaceused
sp_dropremotelogin	sp_syntax
sp_dropsegment	sp_unbindefault
sp_dropserver	sp_unbindmsg
sp_droptreshold	sp_unbindrule
sp_estspace	sp_volchanged
sp_extendsegment	sp_who
sp_foreignkey	sp_column_privileges
sp_help	sp_databases
sp_helpconstraint	sp_datatype_info
sp_helpdb	sp_server_info
sp_helpdevice	sp_table_privileges
sp_helpgroup	

- ◆ **index_type and index_owner columns removed from SYSINDEX system view** The index_type and index_owner columns have been removed from the SYSINDEX view. These columns previously contained the default values USER and SA, respectively. Index information is now stored in the ISYSIDX and ISYSIDXCOL system views. See “[SYSIDX system view](#)” [*SQL Anywhere Reference*, page 755] and “[SYSIDXCOL system view](#)” [*SQL Anywhere Reference*, page 756].
- ◆ **DLL protocol option removed on server** The DLL protocol option now applies only to clients running on Windows 32-bit platforms. The DLL protocol option has been removed from the database server as it uses only Winsock 2.2. Similarly, the DLL protocol has been removed from Windows CE clients as they use only Winsock 1.1.
Winsock 2.2 is required for database servers on all Windows platforms. See “[DLL protocol option](#)” [*SQL Anywhere Database Administration Guide*, page 237].
- ◆ **ASANY and ASANYSH environment variables renamed** The ASANY and ASANYSH environment variables have been renamed SQLANY10 and SQLANYSH10, respectively. See “[SQLANY10 environment variable](#)” [*SQL Anywhere Database Administration Guide*,

page 270] and “SQLANYSH10 environment variable” [*SQL Anywhere Database Administration Guide*, page 270].

- ◆ **Deprecated database properties** The following database properties have been deprecated in this release and always return the value On when their setting is queried:
 - ◆ BlobArenas
 - ◆ SeparateForeignKeys
 - ◆ VariableHashSize
 - ◆ TableBitMaps
 - ◆ FreePageBitMaps
 - ◆ SeparateCheckpointLog
 - ◆ Histograms
 - ◆ LargeProcedureIDs
 - ◆ PreserveSource
 - ◆ TransactionsSpanLogs
 - ◆ CompressedBTrees
- ◆ **Unsupported system procedures** The sa_conn_properties_by_name and sa_conn_properties_by_conn system procedures are no longer supported. You can use the new sa_conn_options system procedure to obtain this information. See “sa_conn_options system procedure” [*SQL Anywhere Reference*, page 844].

MobiLink

Following is a list of additions to MobiLink introduced in version 10.0.0.

New features

Main MobiLink new features

Enhancements to the
MobiLink plug-in in
Sybase Central

It is now much easier to set up MobiLink applications by using a wizard to create a **synchronization model file**. This file contains information you enter about remote and consolidated tables and how to synchronize them. When the model is ready, you can use another wizard to deploy it, which will generate scripts and tables required for the application.

- ◆ **Create Synchronization Model wizard** With the new Create Synchronization Model wizard, you can quickly create and deploy MobiLink applications. This wizard can create a remote database or use existing remotes. It automates the creation of synchronization scripts, and can automatically handle download deletes, conflict resolution, and other challenging synchronization issues.

☞ See “[Error! missing destination]” [\[Getting Started with MobiLink, page ??\]](#).

- ◆ **Model mode** After using the Create Synchronization Model wizard, you can use Model mode to customize your synchronization project before it is deployed. When you are in Model mode, you are working offline. Model mode stores your synchronization model as an XML file.

☞ See “[Error! missing destination]” [\[Getting Started with MobiLink, page ??\]](#).

- ◆ **Deploy wizard** When your model is customized, you can deploy it using the new Deploy wizard. The Deploy wizard adds the scripts, users, script versions, and so on to the MobiLink system tables on the consolidated database. Any changes you make to the consolidated database cannot be reengineered back to Model mode, although you can deploy the same model multiple times.

☞ See “[Error! missing destination]” [\[Getting Started with MobiLink, page ??\]](#).

- ◆ **Admin mode** The MobiLink plug-in as it existed before version 10.0.0 is now called Admin mode. Numerous enhancements have been made to Admin mode to make it easier to use. When you are in Admin mode, you are connected to your consolidated database and making changes live. You can use Admin mode to modify all your MobiLink consolidated databases.

Synchronize to any data source

A new feature called **object-based data flow** allows you to synchronize to virtually any data source. For example, you can synchronize to application servers, web servers, web services, text files, Excel spreadsheets, J2ME devices, or an RDBMS that cannot be used as a consolidated database (such as MySQL). You must still have a consolidated database to hold MobiLink system tables and data that you want MobiLink to manage. The new data source or sources can be fully integrated into your synchronization process.

☞ See “[Synchronizing Data Sources Other than Consolidated Databases](#)” [*MobiLink Administration Guide*, page 559].

In addition, a new feature called **Mobile Web Services** provides support for mobile-optimized asynchronous web services that you can integrate with remote applications.

☞ See “[QAnywhere Mobile Web Service Client Applications](#)” [*QAnywhere User's Guide*, page 157].

Consolidated databases

◆ Setup script required by SQL Anywhere consolidated databases

You must now run a setup script before using a SQL Anywhere database as a MobiLink consolidated database, and the MobiLink system tables that are created with the setup script are now owned by the user who ran the setup script. This behavior is consistent with other consolidated databases. In previous versions of MobiLink, MobiLink system tables were owned by DBO in SQL Anywhere consolidated databases.

☞ See “[SQL Anywhere consolidated database](#)” [*MobiLink Administration Guide*, page 9].

◆ Simplified setup scripts

Each type of consolidated database now has only one setup script. There are no more version-specific setup scripts (such as *syncase125.sql*).

☞ See “[MobiLink Consolidated Databases](#)” [*MobiLink Administration Guide*, page 3].

◆ New ways to clean up MobiLink system tables on your consolidated database

New system procedures have been added that help you do the following:

- Purge information about obsolete remote databases from the MobiLink system tables.

☞ See “[ml_delete_sync_state_before](#)” [*MobiLink Administration Guide*, page 579].

- Delete unused or unwanted synchronization state information.

☞ See “ml_delete_sync_state” [*MobiLink Administration Guide*, page 578].

- Reset synchronization state information.

☞ See “ml_reset_sync_state” [*MobiLink Administration Guide*, page 581].

◆ **New MobiLink system tables and schema** Following are changes to the MobiLink system tables:

- Several new MobiLink system tables have been added. See:
 - ◆ “ml_database” [*MobiLink Administration Guide*, page 600]
 - ◆ “ml_column” [*MobiLink Administration Guide*, page 598]
 - ◆ “ml_qa_clients” [*MobiLink Administration Guide*, page 608]
- The contents of ml_subscription are significantly different. The UltraLite synchronization sequence number, previously stored in ml_user.commmmit_state, is now stored in ml_subscription.progress. The progress column also stores the SQL Anywhere remote synchronization progress.
See “ml_subscription” [*MobiLink Administration Guide*, page 627].
- The contents of ml_user are significantly different.
See “ml_user” [*MobiLink Administration Guide*, page 631].
- A checksum column has been added to the ml_script table.
See “ml_script” [*MobiLink Administration Guide*, page 623].
- The ml_user column of ml_listening has been changed to the name column.
See “ml_listening” [*MobiLink Administration Guide*, page 605].
- A new system table has been added that is used internally by Sybase Central for server-initiated synchronization.
See “ml_sis_sync_state” [*MobiLink Administration Guide*, page 626].
- There have been changes to several QAnywhere system tables. See:
 - ◆ “ml_qa_delivery” [*MobiLink Administration Guide*, page 609]
 - ◆ “ml_qa_global_props” [*MobiLink Administration Guide*, page 611]
 - ◆ “ml_qa_repository” [*MobiLink Administration Guide*, page 614]
 - ◆ “ml_qa_repository_props” [*MobiLink Administration Guide*, page 617]

ODBC driver enhancements

- ◆ **New iAnywhere Solutions Oracle driver** There is a new iAnywhere Solutions Oracle driver that is much easier to configure than the previous driver. The iAnywhere Solutions Oracle driver that was provided with previous versions has been removed.

☞ See “iAnywhere Solutions Oracle driver” [*MobiLink Administration Guide*, page 649].

☞ See also: “Changes to ODBC drivers used by MobiLink” on page 89.

MobiLink synchronization server

- ◆ **Named script parameters** There are now names for MobiLink event parameters. Previously, you had to specify script parameters as question marks. Now, question marks are optional. You can choose from a set of predefined named parameters, or create your own, or both. User-defined named parameters are useful when your RDBMS does not support variables. You can specify the named parameters in any order, and use any subset of available parameters, unlike question marks. Also, in most cases you can use the same named parameter multiple times in the same script.

☞ See “Script parameters” [*MobiLink Administration Guide*, page 257].

- ◆ **Improved MobiLink network layer** The network layer now includes compression, persistent connections (so you can synchronize multiple times on the same connection), resumable connections, IPv6 support, and improved error detection, liveness detection, and debugging.
- ◆ **Improved handling of cache** The MobiLink synchronization server no longer maintains separate pools of memory for different tasks. All cache memory is shared by all synchronizations. You set the cache size using the new `mlsrv10 -cm` option. Other options for setting cache sizes (`-bc`, `-d`, `-dd`, and `-u`) have been removed.

☞ See “-cm option” [*MobiLink Administration Guide*, page 45].

- ◆ **New syntax for `mlsrv10 -x`** The `mlsrv10 -x` option, used for setting network protocol options for MobiLink clients, has changed. To connect the MobiLink server to version 8 or 9 clients, you should use the `mlsrv10 -xo` option, which is identical to the `dbmlsrv9 -x` option. You can support version 8 and 9 clients, as well as version 10 clients, from one instance of `mlsrv10`, but to do so you need to open two ports.

See “-x option” [*MobiLink Administration Guide*, page 88] and “-xo option” [*MobiLink Administration Guide*, page 92].

- ◆ **New conflict detection event** There is a new event that you can script to detect conflicts at the column level. This is an alternative to the `upload_fetch` event, which detects conflicts at the row level.

☞ See “upload_fetch_column_conflict table event” [*MobiLink Administration Guide*, page 444].

- ◆ **Snapshot isolation** For SQL Anywhere version 10 and Microsoft SQL Server 2005 and up consolidated databases, snapshot isolation is

now the default for downloads, and is an option for uploads. MobiLink synchronization server options are added to help you control this behavior.

See:

- ◆ “MobiLink isolation levels” [[Getting Started with MobiLink](#), page 26]
- ◆ “-dsd option” [[MobiLink Administration Guide](#), page 51]
- ◆ “-dt option” [[MobiLink Administration Guide](#), page 52]
- ◆ “-esu option” [[MobiLink Administration Guide](#), page 54]

- ◆ **Forcing upload scripts** The `mlsrv10 -zus` option allows you to force the MobiLink synchronization server to call upload scripts for a table, even when there is no data to upload for that table.
☞ See “-zus option” [[MobiLink Administration Guide](#), page 102].

- ◆ **New verbosity option** The new verbosity option `e` allows you to capture system event scripts. When `-ve` is specified, the MobiLink synchronization server shows all system event scripts that are used to maintain MobiLink system tables, as well as the SQL statements that define the upload stream.
☞ See “-v option” [[MobiLink Administration Guide](#), page 83].

- ◆ **File transfer directory** A new option has been added that allows you to use a directory for file transfers.
☞ See “-ftr option” [[MobiLink Administration Guide](#), page 59].

- ◆ **Limit concurrent network connections** The new `-nc` option lets you specify a limit to the number of concurrent network connections.
☞ See “-nc option” [[MobiLink Administration Guide](#), page 63].

- ◆ **Synchronization ID** Each synchronization is now identified by an integer that is between 1 and 4294967295. Each instance of the MobiLink synchronization server maintains its own synchronization IDs. When the MobiLink synchronization server is started, the ID is reset to 1. This ID is logged in the output file.

- ◆ **Global script version** You can now create a global script version. You define the scripts associated with the global script version once and then they are automatically used in all synchronizations unless you specify a script for the same event in the script version you are using to synchronize. When you are using multiple script versions, this means that you can avoid duplicating connection level scripts.
☞ See “`ml_global script version`” [[MobiLink Administration Guide](#), page 260].

Performance
enhancements

- ◆ **mlsrv10 now uses ISO 8601 datetime format for message timestamps** Timestamps in informational, warning, and error messages now use the unambiguous ISO 8601 datetime format: `{T|W|E} yyyy-mm-dd hh:mm:ss message`.

- ◆ **Improved MobiLink architecture** The MobiLink synchronization server has been re-architected to improve throughput, flexibility, and maintainability. The internal MobiLink client/server protocol has been enhanced for the same reasons.

☞ See “[MobiLink Performance](#)” [*MobiLink Administration Guide*, page 149].

MobiLink Monitor
enhancements

- ◆ **Utility name change to mlmon** The MobiLink Monitor is now called mlmon. Previously, it was called dbmlmon.

☞ See “[Starting the MobiLink Monitor](#)” [*MobiLink Administration Guide*, page 163].

- ◆ **Multiple MobiLink Monitors** You can now connect multiple MobiLink Monitors to the same MobiLink synchronization server simultaneously. This allows multiple users to track synchronizations on the same server.

☞ See “[Starting the MobiLink Monitor](#)” [*MobiLink Administration Guide*, page 163].

- ◆ **Network options** The MobiLink Monitor now allows the same network options as MobiLink clients.

☞ See “[Starting the MobiLink Monitor](#)” [*MobiLink Administration Guide*, page 163].

- ◆ **New Utilization Graph** The Utilization Graph pane provides Monitor statistics in graphical format.

☞ See “[Utilization Graph pane](#)” [*MobiLink Administration Guide*, page 167].

- ◆ **Viewing data in the Chart pane** In the Chart pane, you can still view data by user, or you can choose to view it in Compact view, which shows all active synchronizations in as few rows as possible. The Worker view has been removed.

☞ See “[Chart pane](#)” [*MobiLink Administration Guide*, page 167].

- ◆ **New Sample Properties dialog** The new Sample Properties shows data for a one-second interval or the average of all the one-second intervals in the selected period.

☞ See “[Sample properties](#)” [*MobiLink Administration Guide*, page 169].

- ◆ **Enhanced Session Properties dialog** Session properties now contains a detailed Statistics tab.

☞ See “Session properties” [*MobiLink Administration Guide*, page 169].

MobiLink Redirector enhancements

- ◆ **HTTPS support for Apache Redirector** In previous versions of SQL Anywhere, and in this version for non-Apache Redirectors, when HTTPS is used for the connection between a remote database and web server, the web server decrypts the HTTPS and sends HTTP to MobiLink via the Redirector. Now, for Apache web servers, the Redirector re-encrypts the stream as HTTPS and sends it to the MobiLink server. There is new syntax for the ML directive in the Redirector configuration file.

☞ See “Configuring Redirector properties (all versions)” [*MobiLink Administration Guide*, page 187].

Unix/Linux enhancements

- ◆ **Console** Unix and Linux installations now have a GUI console that shows log information for dbmlsync and mlsrv10.

☞ See “-ux option” [*MobiLink Clients*, page 191] and “-ux option” [*MobiLink Administration Guide*, page 82].

- ◆ **More consistent character conversions** There are improvements to the consistency of character conversions between Unix/Linux and Windows.

MobiLink clients

- ◆ **New remote ID** MobiLink now generates a unique ID called a remote ID the first time a remote database synchronizes (or any time it encounters a NULL value for the remote ID). The remote ID is created automatically as a GUID, but you can set it to any string that has meaning to you. The remote ID is a useful way to identify the remote database when you want a MobiLink user to synchronize multiple remote databases. In UltraLite remote databases, the remote ID is also useful for multiple MobiLink users to synchronize the same remote database.

Every script that accepts the MobiLink user name as a parameter now also accepts a remote_id parameter. The remote_id parameter is only available if you use named parameters. You can still use the MobiLink user name to reference remote databases, but the uniqueness requirement for MobiLink user names is removed.

To help you change the remote ID, a new database option is added to both SQL Anywhere and UltraLite databases called ml_remote_id.

See:

- ◆ “Remote IDs” [*MobiLink Clients*, page 15]
- ◆ “MobiLink user names and remote IDs” on page 87
- ◆ SQL Anywhere clients: “Setting the remote ID” [*MobiLink Clients*, page 15]
- ◆ UltraLite clients: “ml_remote_id option” [*UltraLite Database User's Guide*, page 99]

- ◆ **New file transfer functionality** New functionality helps you transfer files to remote devices on the same connection you use to synchronize data. SQL Anywhere clients can use the new mlfiletransfer utility, and UltraLite clients can use the new MLFileTransfer method. This functionality is especially useful when populating new remote databases or upgrading software. A new MobiLink event has been added to authenticate the file transfer, if desired.

☞ See:

- ◆ SQL Anywhere clients: “MobiLink file transfer utility [mlfiletransfer]” [*MobiLink Clients*, page 31]
- ◆ UltraLite clients: “Using MobiLink file transfers” [*MobiLink Clients*, page 363]
- ◆ MobiLink server: “authenticate_file_transfer connection event” [*MobiLink Administration Guide*, page 294]

- ◆ **Simplified liveness timeout settings** Liveness timeout is now controlled by the client. A new network protocol option called timeout is introduced that replaces liveness_timeout, contd_timeout, unknown_timeout, and network_connect_timeout.

☞ See “timeout” [*MobiLink Clients*, page 58].

- ◆ **Buffer_size enhancements** Using the buffer_size network protocol option, you can now control write buffering for TCP/IP protocols as well as HTTP body size for the HTTP protocols. The default values have also changed.

☞ See “buffer_size” [*MobiLink Clients*, page 35].

UltraLite clients

- ◆ **Palm support for network_leave_open** On Palm devices you can now choose whether network connectivity stays open after synchronization finishes. This functionality was available on other platforms in previous releases.

☞ See “network_leave_open” [*MobiLink Clients*, page 51].

- ◆ **UltraLite enhancements** For information on other UltraLite enhancements, see “Synchronization” on page 103.

SQL Anywhere clients

- ◆ **Scripted upload** In regular synchronization, dbmlsync uses the transaction log to create the upload, and so synchronizes all relevant data that has changed on the remote database since the last upload. You can now write stored procedures that define exactly what rows get uploaded, and so bypass the use of the transaction log. These stored procedures can perform DML and upload the result set, so the rows can be created dynamically, if required.

☞ See “Scripted Upload” [*MobiLink Clients*, page 321].

Support for scripted uploads has required the following changes to SQL Anywhere system objects:

- New column (sync_type) in the ISYSPUBLICATION system table.
☞ See “SYSPUBLICATION system view” [*SQL Anywhere Reference*, page 770].
- New catalog objects in the ISYSSYNCSCRIPT system table for tracking synchronization scripts.
☞ See “SYSSYNCSCRIPT system view” [*SQL Anywhere Reference*, page 782].
- New system procedures convert progress values.
See “sa_convert_ml_progress_to_timestamp system procedure” [*SQL Anywhere Reference*, page 846] and
“sa_convert_timestamp_to_ml_progress system procedure” [*SQL Anywhere Reference*, page 846].

- ◆ **New scheduling options for dbmlsync** When using the EVERY and INFINITE scheduling options, you can now specify that a synchronization does not occur when dbmlsync starts.

☞ See “NoSyncOnStartup (nss) extended option” [*MobiLink Clients*, page 149].

- ◆ **Download-only publications** You can now create publications that only download data. Download-only publications do not use a log file.

☞ See “Download-only publications” [*MobiLink Clients*, page 83].

- ◆ **Error handling enhancements** New event hooks have been added that allow you to process errors reported by dbmlsync on the client.

See:

- ◆ “Handling errors and warnings in event hook procedures” [*MobiLink Clients*, page 206]
- ◆ “sp_hook_dbmlsync_all_error” [*MobiLink Clients*, page 210]
- ◆ “sp_hook_dbmlsync_communication_error” [*MobiLink Clients*, page 216]

- ◆ “[sp_hook_dbmlsync_misc_error](#)” [*MobiLink Clients*, page 250]
- ◆ “[sp_hook_dbmlsync_sql_error](#)” [*MobiLink Clients*, page 267]
- ◆ **Stop dbmlsync from enforcing table order** By default, dbmlsync issues an error if a child table is uploaded before a parent table. A new extended option allows you to override this behavior.
 - ☞ See “[TableOrderChecking \(toc\) extended option](#)” [*MobiLink Clients*, page 157].
- ◆ **New way to track synchronizations** For SQL Anywhere remote databases only, you can now specify a `subscription_id` parameter in your `begin_publication` or `end_publication` script. This value is called `sync_id` in the SYSSYNC system table. This is an advanced feature that helps you track information about your synchronizations.
 - ☞ See “[begin_publication connection event](#)” [*MobiLink Administration Guide*, page 321] and “[end_publication connection event](#)” [*MobiLink Administration Guide*, page 370].
- ◆ **dbmlsync now uses ISO 8601 datetime format for message timestamps** Timestamps in informational, warning, and error messages now use the non-ambiguous ISO 8601 datetime format:




```
{ I | W | E } yyyy-mm-dd hh:mm:ss message.
```
- ◆ **Expanded values in #hook_dict** The dbmlsync utility exposes hooks and passes values as name/value pairs through a temporary table called `#hook_dict`. In the past, the values in the `#hook_dict` table were defined as `VARCHAR(255)`. This has been increased to `VARCHAR(10240)`.

Security

- ◆ **RSA now included with SQL Anywhere** You no longer have to purchase a separate license to use RSA encryption.
 - ☞ See “[Separately licensed components](#)” [*Introducing SQL Anywhere 10*, page 13].
- ◆ **New mlsrv10 -fips option** You can now specify `-fips` when you start the MobiLink server, and thus force all secure connections to use FIPS-approved algorithms. This setting does not affect nonsecure streams.
 - ☞ See “[-fips option](#)” [*MobiLink Administration Guide*, page 57].
- ◆ **FIPS security is supported on more platforms** FIPS security is now supported on more platforms. For a list of supported platforms, see [components_platform_10.html#SA_SeparatelyLicensed](#), located in the `support` subdirectory of your SQL anywhere installation.

- ◆ **Simplified way to specify security streams** The syntax for specifying security options has been simplified on both the server and the client by treating security options as separate network protocols. The following protocols are now supported: TCP/IP, TLS (which is TCP/IP over TLS), HTTP, and HTTPS. The UltraLite Security parameter is removed.
See:
 - ◆ MobiLink server: “-x option” [*MobiLink Administration Guide*, page 88]
 - ◆ SQL Anywhere client: “CommunicationAddress (adr) extended option” [*MobiLink Clients*, page 127]
 - ◆ UltraLite client: “Stream Type synchronization parameter” [*MobiLink Clients*, page 398]
- ◆ **Integration with operating system** By default, MobiLink clients now trust certificates that are already trusted by the operating system on which they operate.

Server-initiated synchronization

- ◆ **Server-initiated synchronization is much easier to set up**
Enhancements have been made to make it much quicker to set up a server-initiated synchronization application:
 - **Sybase Central support** Notifiers and Listeners can now be set up in Sybase Central Model mode, allowing a subset of useful Notification services. In Model mode, you identify a table for server-initiated synchronization, and your download_cursor is automatically used to determine what data is used for notification purposes. When data identified in your download cursor changes, a Notification is sent. The Deployment wizard generates a corresponding Listener options file.
 - **New default gateway** A new gateway called the SYNC gateway allows you to make a persistent connection over the same type of communication path you use for MobiLink synchronization. The SYNC gateway is now the default device tracker gateway, meaning that notification will first try the SYNC gateway, with fallback to the UDP and then SMTP gateways.
 See “Gateways and carriers” [*MobiLink Server-Initiated Synchronization User’s Guide*, page 20].
- ◆ **Shared connections** Multiple Notifiers can now share the same database connection, reducing contention and required server resources in the consolidated database.
 See “shared_database_connection property” [*MobiLink Server-Initiated Synchronization User’s Guide*, page 74].

- ◆ **Persistent connections** The Windows Listener now supports persistent connections. By default, the Listener now maintains a persistent connection to the MobiLink server for device tracking, notification, and confirmation. This feature provides significant performance enhancement over previous versions. It can be disabled with the `dblsn -pc` option.

☞ See “[Listener syntax](#)” [*MobiLink Server-Initiated Synchronization User's Guide*, page 42].

- ◆ **Notifier uses character set of remote device** Notifications are now sent to the remote device using the character set of the remote device. Device tracking information is translated before being applied to the consolidated database.

- ◆ **Custom confirmation handling** You can now implement a Notifier property in SQL that processes the confirmation of a push request and returns its status.





See “[confirmation_handler property](#)” [*MobiLink Server-Initiated Synchronization User's Guide*, page 64].

- ◆ **Custom error handling** You can now implement a Notifier property in SQL that processes errors such as when a push request is not delivered, not confirmed, or improperly confirmed.

☞ See “[error_handler property](#)” [*MobiLink Server-Initiated Synchronization User's Guide*, page 69].

- ◆ **New or changed Windows Listener options** The Listener now supports the following options:


Option	Description
-ni	Stop tracking UDP addresses when <code>-x</code> is used. Previously, this was called <code>-g</code> .
-pc{+ -}	Enable/disable persistent connection for notifications.
-ns	Disables default SMS listening on Windows Mobile 2003 and up Phone Edition.
-nu	Disable default UDP listening.
-r	Register the remote ID file for use by the <code>\$remote_id</code> variable.
-v	When set to 1 or above, the verbosity option now displays and logs command line options.

- ◆ **Remote ID file** On the Listener command line, you can now access the new MobiLink remote ID (which by default is a GUID) using a remote ID file. You do this with the new dblsn option -r and new Listener action variable \$remote_id.
 See “Listener syntax” [*MobiLink Server-Initiated Synchronization User’s Guide*, page 42] and “Action variables” [*MobiLink Server-Initiated Synchronization User’s Guide*, page 50].
- ◆ **New Listener action variables for authentication** There are new action variables that are useful in message handlers: \$ml_user and \$ml_password.
 See “Action variables” [*MobiLink Server-Initiated Synchronization User’s Guide*, page 50].
- ◆ **New Listener action variable for connection parameters** The new \$ml_connect action variable expands to the MobiLink connection parameters that were specified with the dblsn -x option.
 See “Action variables” [*MobiLink Server-Initiated Synchronization User’s Guide*, page 50].
- ◆ **Listener now uses ISO 8601 datetime format for message timestamps** Timestamps in informational, warning, and error messages now use the unambiguous ISO 8601 datetime format: {**T**|**W**|**E**} *yyyy-mm-dd hh:mm:ss message*.
- ◆ **Listener can use TLS** The Listener can now connect to the MobiLink synchronization server using all the network choices as other MobiLink clients. This allows you to apply security to device tracking and notification.
 See -x in “Listener syntax” [*MobiLink Server-Initiated Synchronization User’s Guide*, page 42].
- ◆ **Support for Treo 600, Treo 650, and Kyocera 7135 smartphones** The Palm Listener now supports Treo 600, Treo 650, and Kyocera 7135 smartphones.
For a list of supported platforms, see http://www.ianywhere.com/products/supported_platforms.html.
- ◆ **CE Phone Edition support** The Listener now supports Windows Mobile 2003 Phone Edition for SMS.

Behavior changes and deprecated features

Following is a list of changes to MobiLink introduced in version 10.0.0.

MobiLink synchronization server changes

- ◆ **Cursor-based uploads removed** The following scripts were deprecated in version 9.0.0 and are now removed: `upload_cursor`, `new_row_cursor`, and `old_row_cursor`. You should use statement-based scripts.
 See “Writing scripts to upload rows” [*MobiLink Administration Guide*, page 265].
- ◆ **Unrecognized scripts cause the synchronization to fail** If the MobiLink synchronization server encounters any unrecognized table-level or connection-level scripts, it will abort the synchronization. In previous versions, unrecognized scripts only resulted in a warning message. This means that the presence of cursor-based upload scripts cause the synchronization to abort.
- ◆ **Errors in upload or download scripts cause the synchronization to fail** The synchronization now always aborts if the MobiLink synchronization server encounters errors with upload or download scripts. Previously, the MobiLink synchronization server did not always abort the synchronization.
- ◆ **The `handle_error` and `handle_odbc_error` events work in a more restricted fashion** The `handle_error` and `handle_odbc_error` scripts are now only called when an ODBC error occurs while MobiLink is processing an insert, update, or delete script during the upload transaction, or is fetching download rows. If an ODBC error occurs at another time, the MobiLink synchronization server will call the `report_error` or `report_odbc_error` script and abort the synchronization.
- ◆ **When a begin script is called, its end script is called regardless of the success of the synchronization** There are several MobiLink scripts that have a begin and end form, such as `begin_connection` and `end_connection`. In the past, the end script was often not executed if the synchronization failed. Now, if the begin script is called, the end script is always called (if it is defined), even if the synchronization has errors.
- ◆ **Upload scripts are not called for a table when there is no data to upload** In previous versions, you could use the `-us` option to prevent the MobiLink server from calling upload scripts when there is no data to upload. The `-us` option is now removed and by default, the MobiLink synchronization server only invokes upload scripts when the upload stream contains data to upload. You can revert to the old behavior using the `-zus` option.
See “`-zus` option” [*MobiLink Administration Guide*, page 102].

- ◆ **SQL Anywhere 10 and Microsoft SQL Server 2005 consolidated databases should not change isolation level in the begin_connection script** For SQL Anywhere version 10 and Microsoft SQL Server 2005 and up, the default isolation level for downloads is now snapshot. This means that the isolation level may be changed at the beginning of the download transaction, in which case any setting from the begin_connection script is overridden. Therefore, you should change the isolation level for downloads in the begin_download script or use the new mlsrv10 -dsd option to disable snapshot isolation. Previous documentation recommended changing the isolation level in the begin_connection script, and this is still good practice for consolidated databases that do not use snapshot isolation.

See “[MobiLink isolation levels](#)” [*Getting Started with MobiLink*, page 26]

- ◆ **Options for setting cache size are removed** The following mlsrv10 options have been removed:
 - ◆ -bc
 - ◆ -d
 - ◆ -dd
 - ◆ -u

These options have been replaced with the mlsrv10 -cm option, which sets the cache for all synchronizations.

☞ See “[-cm option](#)” [*MobiLink Administration Guide*, page 45].

- ◆ **Options for setting timeout are removed** The following mlsrv10 options are no longer required and have been removed:
 - ◆ contd_timeout
 - ◆ unknown_timeout

The mlsrv10 liveness_timeout option has also been removed. It is replaced by the timeout option for synchronization clients.

☞ See “[timeout](#)” [*MobiLink Clients*, page 58].

- ◆ **Backlog option no longer required** The mlsrv10 backlog option is no longer required and has been removed.
- ◆ **Changes to protocol names and options for network security** The following network protocol keywords have been removed: https_fips, rsa_tls, rsa_tls_fips, ecc_tls; as well as the network protocol option security. The protocols are not removed, but you specify them differently now. The mlsrv10 -x syntax has changed as follows:

Old syntax	New syntax in version 10.0.0	Description
-x https_fips	-x https(fips=y;...)	HTTPS with FIPS
-x rsa_tls	-x tls(tls_-type=rsa;...)	TCP/IP with TLS using RSA encryption
-x rsa_tls_fips	-x tls(tls_-type=rsa;fips=y;...)	TCP/IP with TLS using RSA encryption and FIPS
-x ecc_tls	-x tls(tls_-type=ecc;...)	TCP/IP with TLS using ECC encryption
-x tcpip(security=...)	-x tcpip	TCP/IP
-x http(security=...)	-x http	HTTP

See “-x option” [*MobiLink Administration Guide*, page 88].

- ◆ **Change to -bn option** The mlsrv10 -bn option compares BLOB bytes during conflict detection. Previously, characters were compared for data of type LONGVARCHAR. Now the units that are compared are always bytes for both binary and LONGVARCHAR BLOBs.

☞ See “-bn option” [*MobiLink Administration Guide*, page 43].

- ◆ **Change to verbosity output** The mlsrv10 options -vr, -vt, and -vu all output slightly different information:
 - **-vr** Now, -vr returns only the upload and download row values. Previously, the upload and download script names and contents were also returned.
 - **-vt** Now, -vt returns only the contents of translated scripts. Previously, the original script contents were also returned.
 - **-vu** Now, -vu returns all undefined table scripts when the scripts need to be invoked. This includes statistical scripts.

See “-v option” [*MobiLink Administration Guide*, page 83].

- ◆ **MobiLink server options -za and -ze are removed** Automatic script generation provided by the MobiLink synchronization server -za option and -ze options has been removed. You can now generate scripts using the Create Synchronization Model wizard.

☞ See “[Error! missing destination]” [*Getting Started with MobiLink*, page ??].

- ◆ **example_upload_cursor, example_upload_delete, example_upload_insert, and example_upload_update table events**

are removed As a result of removing the -za and -ze MobiLink server options, the example_upload_cursor, example_upload_delete, example_upload_insert, and example_upload_update table events are no longer generated. You can now generate scripts using the Create Synchronization Model wizard.

☞ See “[Error! missing destination]” [Getting Started with MobiLink, page ??].

- ◆ **MobiLink server option -oy is removed** The mlsrv10 -oy option, which showed the year in timestamps, has been removed. The year is now always shown in timestamps in informational, warning, and error messages.
- ◆ **Authentication scripts committed** If there is no error, the MobiLink synchronization server always commits the transaction after invoking an authenticate_user, authenticate_user_hashed, or authenticate_parameter script, even if the authentication fails. Previously, transactions involving failed authentication were rolled back, so there could be no record of failed attempts to authenticate.
- ◆ **Changes to statistical properties** The following statistical properties have changed. In previous versions of the documentation, the property descriptions did not explain that the properties return different values based on whether you are using them in normal upload mode or in forced conflict mode; this has been corrected.

Statistical property	Description
conflicted_ - deletes	<p>In normal upload mode, this is always zero.</p> <p>In forced conflict mode, it returns the total number of uploaded deletes that were successfully inserted into the consolidated database using the upload_old_row_insert script.</p> <p>Previously, this returned the number of uploaded deletes for which conflicts were detected.</p>
conflicted_ - inserts	<p>In normal upload mode, this is always zero.</p> <p>In forced conflict mode, it returns the total number of upload inserts that were successfully inserted into the consolidated database using the upload_new_row_insert script.</p> <p>Previously, this returned the number of uploaded inserts for which conflicts were detected.</p>

Statistical property	Description
conflicted_updates	<p>In normal upload mode, this returns the total number of update rows that caused a conflict.</p> <p>In forced conflict mode, it returns the total number of upload update rows that were successfully applied using <code>upload_new_row_insert</code> or <code>upload_old_row_insert</code> scripts.</p> <p>Previously, this returned the number of uploaded updates for which conflicts were detected.</p>
download_bytes	<p>This returns the amount of memory used within the MobiLink synchronization server to store the download.</p> <p>Previously, this returned the number of downloaded bytes.</p>
ignored_deletes	<p>In normal upload mode, this returns the total number of upload delete rows that caused errors while the <code>upload_delete</code> script was invoked, when the <code>handle_error</code> or <code>handle_odbc_error</code> are defined and returned 1000, or when there is no <code>upload_delete</code> script defined for the given table.</p> <p>In forced conflict mode, this returns the total number of upload delete rows that caused errors while the <code>upload_old_row_insert</code> script was invoked, when the <code>handle_error</code> or <code>handle_odbc_error</code> are defined and returned 1000, or when there is no <code>upload_old_row_insert</code> script defined for the given table.</p> <p>Previously, this returned the number of uploaded deletes that were ignored.</p>
ignored_inserts	<p>In normal upload mode, this returns the total number of upload insert rows that caused errors while the <code>upload_insert</code> script was invoked, when the <code>handle_error</code> or <code>handle_odbc_error</code> are defined and returned 1000, or when there is no <code>upload_insert</code> script defined for the given table.</p> <p>In forced conflict mode, this returns the total number of upload insert rows that caused errors while the <code>upload_new_row_insert</code> script was invoked, when the <code>handle_error</code> or <code>handle_odbc_error</code> are defined and returned 1000, or when there is no <code>upload_insert</code> script defined for the given table.</p> <p>Previously, this returned the number of uploaded inserts that were ignored.</p>

Statistical property	Description
ignored_updates	<p>In normal upload mode, this returns the total number of upload update rows that caused errors while the upload_update script was invoked, when the handle_error or handle_odbc_error are defined and returned 1000, or when there is no upload_update script defined for the given table.</p> <p>In forced conflict mode, this returns the total number of upload update rows that caused errors while the upload_new_row_insert or upload_old_row_insert scripts were invoked, or when the handle_error or handle_odbc_error are defined and returned 1000.</p> <p>Previously, this returned the number of uploaded updates that were ignored.</p>
upload_bytes	<p>This returns the amount of memory used within the MobiLink synchronization server to store the upload.</p> <p>Previously, this returned the number of uploaded bytes.</p>
upload_deleted_rows	<p>In normal upload mode, this returns the total number of rows that were successfully deleted from the consolidated database.</p> <p>In forced conflict mode, this is always zero.</p> <p>Previously, this returned the number of row deletions that were uploaded from the synchronization client.</p>
upload_inserted_rows	<p>In normal upload mode, this returns the total number of rows that were successfully inserted in the consolidated database.</p> <p>In forced conflict mode, this is always zero.</p> <p>Previously, this returned the number of row insertions that were uploaded from the synchronization client.</p>
upload_updated_rows	<p>In normal upload mode, this returns the total number of rows that were successfully updated in the consolidated database.</p> <p>In forced conflict mode, this is always zero.</p> <p>Previously, this returned the number of row updates that were uploaded from the synchronization client.</p>

See “MobiLink statistical properties” [*MobiLink Administration Guide*, page 175].

- ◆ **Null characters can now be synchronized to and from columns with CHAR or NCHAR data types in the remote database** Previously in MobiLink, VARCHAR and CHAR column values containing null characters could cause a synchronization to fail. Now you can synchronize null characters in remote database columns of data type CHAR, VARCHAR, LONG VARCHAR, NCHAR, NVARCHAR, AND LONG NVARCHAR.

- ◆ **New format for logging information, warning, and error messages** Previously, the MobiLink synchronization server logged messages in the following format:

```
T.mm/dd hh:mm:ss. thread_id User_name: message
```

Now, the MobiLink server logs messages in the following format:

```
T.mm/dd hh:mm:ss. synchronization_id: message
```

For each synchronization, the first message in the log shows the remote ID, user name, script version, and client name (UltraLite or SQL Anywhere).

The new format reduces the size of the output log without reducing the information that is provided.

- ◆ **New data type in system procedures for Oracle** In MobiLink system procedures that are used to register scripts, the script contents parameter now uses the CLOB data type for Oracle consolidated databases. In the ml_add_property system procedure, the prop_value parameter is now CLOB for Oracle. Previously, these parameters were type VARCHAR.

☞ See “[MobiLink System Procedures](#)” [[MobiLink Administration Guide](#), page 569].

- ◆ **Redirector supports different set of web servers** The MobiLink Redirector has been updated to support newer versions of web servers. In many cases, the older versions may still work. However, the newer versions have been documented and tested with this release.

☞ See [components_platform_10.html#Redirector](#).

MobiLink client changes

MobiLink user names and remote IDs

MobiLink now generates a unique ID called a remote ID the first time a remote database synchronizes (or when it encounters a NULL value for the remote ID). The MobiLink user name no longer needs to be unique. The MobiLink user name can now be considered a true user name that is used for authentication.

In previous versions, the synchronization progress was stored for the MobiLink user name. Now, the progress is stored for the remote ID and subscription for SQL Anywhere remotes, and the remote ID and publication for UltraLite remotes.

☞ See “[ml_subscription](#)” [*MobiLink Administration Guide*, page 627].

You now have the option of using the Mobilink user name or the remote ID to reference remote databases. Previously, you used the MobiLink user name to uniquely identify a remote database. The remote ID is a useful way to identify the remote database when you want a MobiLink user to synchronize multiple remote databases. In UltraLite remote databases, the remote ID is also useful for multiple MobiLink users to synchronize the same remote database.

☞ See “[Remote IDs](#)” [*MobiLink Clients*, page 15].

UltraLite clients

☞ See “[Behavior changes and deprecated features](#)” on page 110.

SQL Anywhere clients

◆ **Download error hooks deprecated** The following error hooks are deprecated: `sp_hook_dbmlsync_download_com_error`, `sp_hook_dbmlsync_fatal_sql_error`, and `sp_hook_dbmlsync_sql_error`. They have been replaced.

☞ See “[Handling errors and warnings in event hook procedures](#)” [*MobiLink Clients*, page 206].

◆ **log_rescan hook only called if dbmlsync expects another synchronization** Previously, the `sp_hook_dbmlsync_log_rescan` hook was called at the end of every synchronization. This caused a pause to occur after dbmlsync disconnected from the MobiLink server, but before the “synchronization complete” message was displayed in the log. Now, the hook is only called when dbmlsync expects another synchronization, for example when the `dbmlsync -n` option is specified more than once in a command line or when scheduling is enabled.

☞ See “[sp_hook_dbmlsync_log_rescan](#)” [*MobiLink Clients*, page 244].

◆ **Liveness timeout options simplified** On the client, the `liveness_timeout` and `network_connect_timeout` network connection protocol options are removed. Use the `timeout` connection option instead.

☞ See “[timeout](#)” [*MobiLink Clients*, page 58].

◆ **Version 7 syntax and utilities are removed** The following SQL statements and utility were deprecated and are now removed:

- MobiLink client database extraction utility (`mlxtract`)
- `CREATE SYNCHRONIZATION SITE` statement
- `CREATE SYNCHRONIZATION DEFINITION` statement

- CREATE SYNCHRONIZATION TEMPLATE statement

Miscellaneous MobiLink behavior changes

Changes to ODBC
drivers used by MobiLink

- ◆ **Sybase Adaptive Server Enterprise driver** SQL Anywhere no longer includes an iAnywhere Solutions ODBC driver for Adaptive Server Enterprise. Instead, the Adaptive Server Enterprise native driver is tested to work with MobiLink. The iAnywhere Solutions 9 - Adaptive Server Enterprise Wire Protocol driver is no longer supported.

☞ See

http://www.iAnywhere.com/developer/technotes/odbc_mobilink.html.

- ◆ **IBM UDB DB2 driver** SQL Anywhere no longer includes an iAnywhere Solutions ODBC driver for DB2. Instead, the IBM DB2 8.2 CLI driver is tested to work with MobiLink. This native DB2 driver supports DB2 versions 8.1 and 8.2. The following drivers are no longer supported: IBM DB2 7.2 ODBC driver and iAnywhere Solutions 9 - DB2 Wire Protocol driver.

☞ See

http://www.iAnywhere.com/developer/technotes/odbc_mobilink.html.

- ◆ **Oracle driver** There is a new iAnywhere Solutions Oracle driver that is easier to configure. It is called iAnywhere Solutions 10 - Oracle. This driver is installed with SQL Anywhere. The iAnywhere Solutions 9 - Oracle Wire Protocol driver is no longer supported.

☞ See

http://www.iAnywhere.com/developer/technotes/odbc_mobilink.html.

Version support

- ◆ **Support for clients prior to version 8.0.0 removed** The MobiLink synchronization server no longer supports SQL Anywhere clients prior to version 8.0.0. To use older databases with the version 10 MobiLink synchronization server, you need to follow upgrade procedures.

☞ See

“Upgrading SQL Anywhere” on page 334.

Name changes

The following utility names have changed:

Old utility name	New utility name
dbmlsrv9	mlsrv10
dbmluser	mluser
dbmlmon	mlmon
dbmlstop	mlstop
dbasinst	mlasinst


The following file names have changed:

Old file name	New file name
<i>dbmlsv9.dll</i>	<i>mlodbc10.dll</i>
<i>dbasdesk.dll</i>	<i>mlasdesk.dll</i>
<i>dbasdev.dll</i>	<i>mlasdev.dll</i>
<i>dbmlsrv.mle</i>	<i>mlsrv10.mle</i>
<i>syncasa.sql</i>	<i>syncsa.sql</i>

Server-initiated
synchronization

- ◆ **Windows SDK removed** The SDK for creating support for more Windows devices has been removed. It is replaced by improved support for SMS. The Palm SDK remains.
- ◆ **Listener -g option is replaced** The dblsn -g option is replaced with the dblsn -ni option.

Other MobiLink behavior
changes

- ◆ **Support for Windows Performance Monitor is dropped** MobiLink no longer supports Windows Performance Monitor. You should use the MobiLink Monitor instead.
 See “MobiLink Monitor” [*MobiLink Administration Guide*, page 161].

QAnywhere

New features

Following is a list of additions to QAnywhere introduced in version 10.0.0.

Mobile web services

Mobile web services provide support for mobile-optimized asynchronous web services. This allows mobile applications to make web service requests—even when they are offline—and have those requests queued for transmission later. The requests are delivered as messages using QAnywhere. A web services connector on the server side takes the client request and forwards it to the web service. It then takes the response from the web service and returns it to the client as a message. A WSDL compiler is provided that facilitates the use of mobile web services from your .NET or Java application.

☞ See “QAnywhere Mobile Web Service Client Applications” [*QAnywhere User's Guide*, page 157].

New QAnywhere plug-in for Sybase Central

Sybase Central now includes a QAnywhere plug-in that provides an easy-to-use graphical interface for creating and administering your QAnywhere applications. With the QAnywhere plug-in, you can:

- ◆ Create client and server message stores.
- ◆ Create and maintain configuration files for the QAnywhere Agent.
- ◆ Browse QAnywhere Agent log files.
- ◆ Create or modify destination aliases.
- ◆ Create JMS connectors and web service connectors.
- ◆ Create and maintain transmission rules files.
- ◆ Browse message stores remotely.
- ◆ Track messages.

Although QAnywhere is not supported on Unix platforms, you can now use Sybase Central on Unix to track messages.

New QAnywhere client APIs

- ◆ **New SQL API** The QAnywhere SQL API is a set of SQL stored procedures that allow SQL developers to easily leverage QAnywhere messaging capabilities. Using this API, stored procedures can send or receive messages using a straightforward approach that complements existing database applications. This can allow for powerful applications that combine database and messaging operations in a single transaction.

For example, a stored procedure could insert a row into the database and send a message to another application—and have both actions committed as part of the same transaction.

☞ See “[QAnywhere SQL API Reference](#)” [*QAnywhere User’s Guide*, page 509].

- ◆ **New Java client API** The new QAnywhere client API for Java helps you create messaging client applications in Java. The client API for Java is currently supported only on Windows.

☞ See “[QAnywhere Java API Reference](#)” [*QAnywhere User’s Guide*, page 415].

QAnywhere client API enhancements

The following additions have been made to the QAnywhere client APIs:

- ◆ **Message selectors** You can now use SQL-like expressions to selectively browse or receive messages from a queue. The syntax for creating message selectors is identical to that used for conditions in transmission rules.

☞ See “[Browsing QAnywhere messages](#)” [*QAnywhere User’s Guide*, page 97].

- ◆ **New ways to browse messages** You can now browse messages from multiple queues, or browse subsets of messages based on ID or message selector.

☞ See “[Browsing messages using a selector](#)” [*QAnywhere User’s Guide*, page 99].

- ◆ **Enumerate message store property names** You can now enumerate message store property names.

☞ See “[Enumerating message store properties](#)” [*QAnywhere User’s Guide*, page 183].

- ◆ **Undeliverable messages** Using the new message store property `ias_MaxDeliveryAttempts`, you can set the maximum number of attempts that a QAnywhere client will attempt to receive a message before considering it undeliverable.

☞ See “[Transmission rule variables](#)” [*QAnywhere User’s Guide*, page 201].

- ◆ **Cancelling messages** You can now cancel messages before they are sent.

☞ See “[Cancelling QAnywhere messages](#)” [*QAnywhere User’s Guide*, page 89].

QAnywhere Agent new features

- ◆ **Query message status** You can now query the status of a message using new pre-defined message properties: `ias_Status` and `ias_StatusTime`. You can also query the originator of a message with `ias_Originator`, or the number of times the message has been delivered to a receiver with `ias_DeliveryCount`.

☞ See “Pre-defined message properties” [*QAnywhere User's Guide*, page 173].

- ◆ **New message store property to set upload increments** `ias_MaxUploadSize` can be used to change the upload increment.

☞ See “Pre-defined message store properties” [*QAnywhere User's Guide*, page 179].

- ◆ **Multiple agents on a single device** Previously, you could only run one instance of the QAnywhere Agent on a device. This limitation has been removed.

☞ See “Running the QAnywhere Agent” [*QAnywhere User's Guide*, page 45].

- ◆ **More options for setting up failover** There are two new QAnywhere Agent options, `-fd` and `-fr`, that help you customize the way failover occurs.

See “`-fd` option” [*QAnywhere User's Guide*, page 128] and “`-fr` option” [*QAnywhere User's Guide*, page 129].

- ◆ **Persistent connections** The new option `-pc+` has been added to enable persistent connections for message transmission. The new `-push` option replaces `-push_notifications` and now allows you to specify whether you want push notifications to use persistent connections.

See:

- ◆ “`-pc` option” [*QAnywhere User's Guide*, page 138]
- ◆ “`-push` option” [*QAnywhere User's Guide*, page 141]

- ◆ **New upgrade procedure** The new `-sur` option can be used to upgrade a client message store from a previous version of SQL Anywhere.

☞ See “`-sur` option” [*QAnywhere User's Guide*, page 148].

- ◆ **QAnywhere Agent now uses ISO 8601 datetime format for message timestamps** Timestamps in informational, warning, and error messages now use the non-ambiguous ISO 8601 datetime format:

```
{ I | W | E } YYYY-mm-dd hh:mm:ss message.
```






Other QAnywhere enhancements

- ◆ **QAnywhere message store can be shared by non-messaging applications** You can now use the same SQL Anywhere database for your application data and as a message store.

- ◆ **Destination aliases** You can now define a destination alias that represents a set of QAnywhere destinations. Messages sent to a destination alias are sent to each member of the alias.
☞ See [“Destination aliases” \[QAnywhere User’s Guide, page 62\]](#).
- ◆ **Server management requests** You can now create server administration requests on the client and send them to the server message store for processing. This includes creating destination aliases or monitoring, starting, and stopping JMS connectors.
☞ See [“Administering the server message store” \[QAnywhere User’s Guide, page 33\]](#).
- ◆ **Improved maintenance of server transmission rules** You can now change the default server transmission rules and the change will automatically be applied to all clients. Previously, to change the default you had to manually define a transmission rule for each client.
☞ See [“Server transmission rules” \[QAnywhere User’s Guide, page 189\]](#).
- ◆ **More message properties** Additional pre-defined message properties are set by QAnywhere, giving you more flexibility in processing messages, better information during debugging, and more help with troubleshooting the status of messages.
☞ See [“Message properties” \[QAnywhere User’s Guide, page 173\]](#).
- ◆ **Ability to embed backslashes in JMS destinations** JMS destinations can now include subcontexts that require backslash delimiters.
☞ See [“Addressing QAnywhere messages meant for JMS” \[QAnywhere User’s Guide, page 113\]](#).
- ◆ **New transmission rule functions** The following transmission rule functions have been added for improved date handling:
 - ◆ DATEADD(datepart, count, datetime)
 - ◆ DATEPART(datepart, date)
 - ◆ DATETIME(string)
 - ◆ LENGTH(string)
 - ◆ SUBSTR(string, start, length)See [“Transmission rule functions” \[QAnywhere User’s Guide, page 199\]](#).
- ◆ **Prefaces for properties in transmission rules** You can now preface message property names and message store property names when you use them in transmission rules and so bypass the precedence given to transmission rule variables of the same name.
☞ See [“Using properties as transmission rule variables” \[QAnywhere User’s Guide, page 201\]](#).

Behavior changes and deprecated features

Following is a list of changes to QAnywhere introduced in version 10.0.0.

- ◆ **qaagent -port is removed** The `-port` option specified a port number on which QAnywhere Agent listened for communications from the Listener. This option is no longer required and has been removed. A free port is automatically used.
- ◆ **qaagent -la_port is replaced** The `-la_port` option has been replaced by the `-lp` option.
 See “[-lp option](#)” [*QAnywhere User's Guide*, page 132].
- ◆ **qaagent -push_notifications is renamed** This option is now called `-push`. It now allows you to enable push notifications with or without persistent connection.
 See “[-push option](#)” [*QAnywhere User's Guide*, page 141].
- ◆ **getPropertyNames** The `getPropertyNames` function has been removed from the C++ client API. It has been replaced with `beginEnumPropertyNames`, `nextPropertyName`, and `endEnumPropertyNames`.
 See “[QAMessage Class](#)” [*QAnywhere User's Guide*, page 387].
- ◆ **Date handling in transmission rules** The following transmission rule message store variables have been removed:
 - ◆ `ias_CurrentDayOfWeek`
 - ◆ `ias_CurrentDayOfMonth`
 - ◆ `ias_CurrentMonth`
 - ◆ `ias_CurrentYear`
 In their place, you can use `ias_CurrentTimestamp` or `DATEPART`.
 See “[Transmission rule variables](#)” [*QAnywhere User's Guide*, page 201].
- ◆ **Transaction log is not used or maintained** The QAnywhere Agent no longer uses a transaction log or manages its size. As a result, for most applications the client message store should be created using the `dbinit -n` option, which initializes the database with no transaction log.
 See “[Setting up the client message store](#)” [*QAnywhere User's Guide*, page 43].
- ◆ **QAnywhere Central replaced** QAnywhere Central has been replaced with the QAnywhere plug-in to Sybase Central. The plug-in provides many enhancements in functionality.

SQL Remote

New features

Following is a list of additions to SQL Remote introduced in version 10.0.0.

- ◆ **invalid_extensions option** A new messaging option has been added that allows you to stop SQL Remote from using certain file extensions in FILE and FTP messaging.

☞ See “[SET REMOTE OPTION statement \[SQL Remote\]](#)” [[SQL Anywhere Reference](#), page 644].

- ◆ **The Message Agent (dbremote) has a graphical user interface on Unix and Linux platforms** See -ux option in “[Message Agent](#)” [[SQL Remote User's Guide](#), page 188].

- ◆ **dbremote now uses ISO 8601 datetime format for message timestamps** Timestamps in informational, warning, and error messages now use the non-ambiguous ISO 8601 datetime format:
`{I|W|E} YYYY-mm-dd hh:mm:ss message.`

Behavior changes and deprecated features

Following is a list of changes to SQL Remote introduced in version 10.0.0.

- ◆ **Support for Adaptive Server Enterprise databases is removed** SQL Remote no longer supports Adaptive Server Enterprise consolidated databases. This means that ssxtract, ssremote, ssqueue, and all other SQL Remote for Adaptive Server Enterprise utilities and files are removed from the install.

To synchronize Adaptive Server Enterprise databases, you should use MobiLink.

☞ For information about upgrading from SQL Remote to MobiLink, see http://www.ianywhere.com/whitepapers/migrate_to_ml.html.

- ◆ **Extract Database wizard no longer extracts legacy databases** The Extract Database wizard only works with version 10 databases.
- ◆ **Expanded values in #hook_dict** The utilities dbxtract and dbremote expose hooks and pass values as name/value pairs through a temporary table called #hook_dict. In the past, the values in the #hook_dict table were defined as VARCHAR(255). This has been increased to VARCHAR(10240).

UltraLite

New features

Following is a list of additions to UltraLite introduced in version 10.0.0.

Main features

UltraLite is now a full-featured relational database management system, designed with ease-of-administration and SQL Anywhere compatibility in mind. Despite the addition of many new and useful features, UltraLite still maintains a small footprint size. See [“UltraLite size and number limitations” \[UltraLite Database User's Guide, page 10\]](#) for a complete list of UltraLite limitations for this release.

Main features of this release include:

- ◆ **Increased database limits** The UltraLite database limits have been dramatically increased. In particular, the maximum number of rows in a table has been increased to 16 million. See [“UltraLite size and number limitations” \[UltraLite Database User's Guide, page 10\]](#) for other current database limits.
- ◆ **Integrated schema** UltraLite is now a standalone RDBMS and no longer requires a separate schema file to define the logical structure of the database. For this release, the UltraLite schema is fully integrated with the database. See [“The UltraLite database schema” \[UltraLite Database User's Guide, page 17\]](#) for details on the internal database schema.
- ◆ **Consolidated file formats** File formats have been consolidated in version 10 of UltraLite. This means that most platforms can now share a database file. If you need characters that are not defined by the collation you require, you should now choose to UTF-8 encode your database. See [“Platform requirements for character set encoding” \[UltraLite Database User's Guide, page 38\]](#) and [“utf8_encoding property” \[UltraLite Database User's Guide, page 115\]](#) for details.
- ◆ **Increased database performance and data integrity** Overall, the UltraLite database performance and data integrity has been improved with several indexing and database page management improvements.
- ◆ **Indexes may utilize hashing** Indexes may now be specified to utilize hashing. The hash size can be specified on a per-index basis. The hash size can improve performance of index lookups and may affect database file size. See [“Indexes in UltraLite databases” \[UltraLite Database User's Guide, page 23\]](#)

- ◆ **Direct database creation** You can now create UltraLite database files directly; a database schema file or reference database file is not required as the source for an UltraLite database. Instead, you can independently create a UltraLite database with Sybase Central or a command line utility, or even programmatically from an application.

For existing UltraLite users, you can no longer create databases in the same manner as previous versions. See [“Upgrading UltraLite” on page 349](#).

- ◆ **Embedded SQL as a dynamic SQL programming interface** In previous versions, embedded SQL was a static interface. In this version, it is an interface to UltraLite dynamic SQL and does not require a SQL Anywhere database. Embedded SQL support also supports dynamic ESQL statements and the use of host variable placeholders. Furthermore, ESQL applications can also now run with `ulengl0`. You can achieve this by linking against `ulrtc.lib` instead of `ulrt.lib`.

As a result of this change, you may notice that simple Embedded SQL applications could grow in size, whereas complex applications may become smaller. See [“Upgrading UltraLite” on page 349](#) and [“Developing embedded SQL applications” \[UltraLite C/C++ User’s Guide, page 4\]](#).

Platforms and devices

Platform support has been modified. For a list of supported platforms, see *components_platform_10.html*, located in the *support* subdirectory of your SQL Anywhere installation.

Important enhancements to note include:

- ◆ **Deployment Platforms** Platform enhancements include:
 - **Palm OS** UltraLite support of Palm OS devices has been enhanced with the following changes:
 - ◆ Runtime Support for Palm OS v4.x and higher.
 - ◆ Development support for CodeWarrior has been increased to version 9. Note that CodeWarrior 8 is no longer supported.
 - ◆ Support for multiple databases and generalized filenames. You can now specify a database for multiple devices with the DBF connection parameter, ensuring that you set the filename correctly depending on whether you are using record-based or file-based storage. See [“file_name connection parameter \[DBF\]” \[UltraLite Database User’s Guide, page 120\]](#) and [“Platform-specific file name options” \[UltraLite Database User’s Guide, page 63\]](#).
 - ◆ Support for both NVFS devices and VFS devices.

- **Symbian OS** Symbian OS support is new to this version of UltraLite. UltraLite supports versions 7.0 and 8.0 of Symbian OS on UIQ phones (2.0 and 2.1) and Nokia S60 (second edition), and Series 80 devices.
- **Windows Mobile 2005** If you are using Embedded Visual C++ 3.0 or 4.0, you can continue to use the existing runtimes. However, new runtimes (installed under `\ultralite\ce\arm.50`) are required when using Visual Studio 2005 to build an application.
- ◆ **Enhanced development environment support** Development tools and languages have been updated as follows:
 - UltraLite now supports ADO.NET 1.0 development in Visual Studio.NET 2003 and ADO.NET 2.0 development in Visual Studio 2005.
 - UltraLite now supports AppForge CrossFire version 5.6 for Visual Basic and C# development. You can deploy applications for AppForge to Palm OS, Symbian OS, and Windows CE platforms
 - C++ component development.

Security

- ◆ **Encryption types** If you are compressing over TLS, UltraLite also supports both ECC and RSA encryption types. RSA encryption is no longer separate product.
- ◆ **FIPS security** You can now secure MobiLink server communications with FIPS security.
- ◆ **Simplified security streams** You can now define encrypted streams as a network protocol or stream type rather than use a separate security parameter. The complete set of supported stream types is: TCP/IP, TLS (for RSA, ECC, and FIPS), HTTP, and HTTPS. See “[Stream Type synchronization parameter](#)” [*MobiLink Clients*, page 398].

Database management

Important new features and enhancements include:

- ◆ **Password changes** All passwords are case-sensitive, regardless of the case-sensitivity of the database. New databases are created with a default user ID of **DBA** with the password **sql**. Consequently user IDs, passwords and trusted root certificates are not preserved as you upgrade your database from earlier releases.
- ◆ **Improved database properties and connection parameters** Database properties and connection parameters have been enhanced and

simplified to allow you describe database and connection behavior more easily. See “[UltraLite Database Settings Reference](#)” [*UltraLite Database User’s Guide*, page 85] and “[UltraLite Connection String Parameters Reference](#)” [*UltraLite Database User’s Guide*, page 117] for a complete list of database properties and connection parameters you can set for this release of the database.

- ◆ **Increased index performance** UltraLite index performance has been enhanced with this release. One of the major improvements is the introduction of index hashing. UltraLite now has a configurable index hash size. By setting the hash size to a value between 1-32 bytes, UltraLite stores part or all of the indexed value in the index page. This reduces the number of row lookups required. See “[Index hashing considerations](#)” [*UltraLite Database User’s Guide*, page 39] .
- ◆ **Checksum validation** You can now include checksums on database pages to validate data integrity of these pages as they are stored on disk. See “[Verifying page integrity with checksums](#)” [*UltraLite Database User’s Guide*, page 46] and “[checksum_level property](#)” [*UltraLite Database User’s Guide*, page 87].
- ◆ **Extended BLOB support** UltraLite databases now have extended support for BLOBs. UltraLite allows you to update, cast data types and get the length of these BLOBs.

Administration tools

Administration tools have been enhanced with this release. To ensure the correct usage of a tool, ensure you review the documentation for it.

Graphical administration tools

- ◆ **Sybase Central** You can now use Sybase Central to create, modify, and administer your UltraLite databases in a graphical user interface. This replaces the ulview schema editing utility.
The list of wizards included in Sybase Central include:
 - Use the Create Database wizard to build a new UltraLite database. This wizard shares the same functionality as the ulcreate utility.
 - Use the Erase Database to erase an existing UltraLite database. No utility equivalent exists for this wizard.
 - Use the Extract Database to initialize a new UltraLite database from a SQL Anywhere reference database. This wizard shares the same functionality as the ulinit utility.
 - Use the Load Database to load an XML file into an UltraLite database. This wizard shares the same functionality as the ulload utility.

- Use the Migrate C/C++ Application to migrate C/C++ formerly created with the removed ulgen utility. No utility equivalent exists for this wizard.
- Use the Synchronize Database wizard to synchronize an UltraLite database. This wizard shares the same functionality as the ulsync utility.
- Use the Upgrade Database to upgrade an existing UltraLite database from a previous version. This wizard shares the same functionality as the ulunloadold utility when used with the ulload utility.
- Use the Unload Database to unload data/schema information from an UltraLite database to XML, SQL, or another database. This wizard shares the same functionality as the ulunload utility with the additional functionality of the ulcreate and ulload utilities.

See [“SQL Anywhere UltraLite Help” \[SQL Anywhere 10 Help, page 199\]](#).

- ◆ **Interactive SQL** You can now use Interactive SQL to develop and test SQL statements with UltraLite databases. Interactive SQL replaces the ulisql utility used in previous versions. See [“Interactive SQL utility \(dbisql\)” \[UltraLite Database User's Guide, page 135\]](#).

Command line administration tools

The following command line utilities are new to UltraLite:

- ◆ **Unload Old Database utility** The new ulunloadold command line utility helps you to unload existing 8.0.2 or 9.x UltraLite databases (schema + data) or schema files to an XML file. With the ulload command line utility, you can then use that output to rebuild an UltraLite version 10 database. See [“UltraLite Unload Old Database utility \(ulunloadold\)” \[UltraLite Database User's Guide, page 169\]](#).
- ◆ **Information utility** The new ulinfo utility displays information about an UltraLite database. It can also change and/or clear database option IDs like global_id or ml_remote_id. See [“UltraLite Information utility \(ulinfo\)” \[UltraLite Database User's Guide, page 151\]](#).

Also, because existing command line utilities have been enhanced to support the new RDBMS features in UltraLite 10, the options for these utilities have changed from earlier versions. To ensure you are using new utilities correctly, ensure you review the reference documentation before starting. See [“UltraLite Utilities Reference” \[UltraLite Database User's Guide, page 133\]](#) for complete utility reference notes.

- ◆ **Enhanced error reporting** UltraLite utilities now report errors consistently with other SQL Anywhere utilities.

- ◆ **Extended database creation options** All database creation utilities (for example, ulcreate and ulload) now support the use of extended creation options. These extended options are configured on the command line with -o, and allow you to configure the same set of database properties that you can set with Sybase Central wizards. See “[Extended creation-time options](#)” [*UltraLite Database User’s Guide*, page 172] for details on how to use extended creation options correctly.
- ◆ **Enhanced unload behavior** You can now use ulunload to output the UltraLite database schema as a sequence of dynamic SQL statements. See “[UltraLite Unload Database to XML utility \(ulunload\)](#)” [*UltraLite Database User’s Guide*, page 166].
- ◆ **Enhanced ulsync behavior** ulsync allows you to set network protocol options and extended synchronization parameters directly from this utility. See “[UltraLite Synchronization Parameters and Network Protocol options](#)” [*MobiLink Clients*, page 381] for a complete list.
Additionally, ulsync now allows you to name publications, not just the publication mask. The keyword **Publications** takes a comma separated list of publication names. See “[UltraLite Synchronization utility \(ulsync\)](#)” [*UltraLite Database User’s Guide*, page 163] for details.
- ◆ **Enhanced conduit installation** The HotSync Conduit Installation utility (ulcond10) now supports conduit extensions, connection strings, and multiple databases. See “[UltraLite HotSync Conduit Installation utility for Palm OS \(ulcond10\)](#)” [*UltraLite Database User’s Guide*, page 142] for details.
- ◆ **ulmbvreg** The ulmbvreg utility that registers UltraLite for AppForge has been renamed to ulafreg. This utility is now installed to the *install-dir\win32* directory. See “[UltraLite Registry update utility \(ulafreg\)](#)” [*UltraLite Database User’s Guide*, page 144].

ULSQLCONNECT

Previously, all UltraLite utilities received connection information from the command line. Now, if you want to pass information other than default user IDs and passwords, you can set the ULSQLCONNECT environment variable on your host machine. See “[Setting the ULSQLCONNECT environment variable](#)” [*UltraLite Database User’s Guide*, page 65].

SQL

- ◆ **SQL Statements** UltraLite supports several new statements. These new statements include:
 - **ALTER TABLE** In addition to creating tables with UltraLite SQL, you can now alter the definition with this statement. See “[ALTER TABLE statement](#)” [*UltraLite Database User’s Guide*, page 284].

- **ALTER/CREATE/DROP PUBLICATION** UltraLite now supports the addition, creation, and deletion of publications with these three statements. See “[ALTER PUBLICATION statement](#)” [*UltraLite Database User's Guide*, page 288], “[CREATE PUBLICATION statement](#)” [*UltraLite Database User's Guide*, page 292] and “[DROP PUBLICATION statement](#)” [*UltraLite Database User's Guide*, page 300].
- **START/STOP SYNCHRONIZATION DELETE** UltraLite SQL includes these statements. Use these statements to control how deletes are logged with MobiLink synchronization. See “[START SYNCHRONIZATION DELETE statement](#)” [*UltraLite Database User's Guide*, page 309] and “[STOP SYNCHRONIZATION DELETE statement](#)” [*UltraLite Database User's Guide*, page 310].
- ◆ **Named constraints** Table constraints can now be named in the ALTER TABLE and CREATE TABLE statements. This permits modification of table and column constraints by changing individual constraints, rather than by modifying an entire table constraint. See “[ALTER TABLE statement](#)” [*UltraLite Database User's Guide*, page 284] and “[CREATE TABLE statement](#)” [*UltraLite Database User's Guide*, page 293].
- ◆ **Other SELECT statement enhancements** The SELECT statement has been extended:
 - SELECT statements can now include START AT as part of the TOP clause. START AT provides additional flexibility in queries that explicitly limit the result set. See “[SELECT statement](#)” [*UltraLite Database User's Guide*, page 304].
 - The DISTINCT clause has been enhanced to allow aggregate functions with this clause (for example, SUM, AVERAGE, MAX, and so on). If you use aggregate functions with this clause, you will significantly increase the execution time. See “[SELECT statement](#)” [*UltraLite Database User's Guide*, page 304] and “[Alphabetical list of functions](#)” [*UltraLite Database User's Guide*, page 223].
- ◆ **UNION operator** The UNION operator allows you to build a single result set from two or more queries into a single result set. By default, the UNION operator removes duplicate rows from the result set. See “[Combining sets with the UNION operation](#)” [*SQL Anywhere SQL User's Guide*, page 323].

Synchronization

- ◆ **Configurable and increased default cache size for HotSync conduit synchronization** Previously, beyond a certain amount of data synchronized on Palm OS file-based data stores, synchronization speeds

were negatively affected. Consequently, the default cache size (on desktop) for the UltraLite conduit has been increased to 4 MB. This increased cache size significantly improves the synchronization time by cutting down unnecessary file I/O operations. However, you can also configure a different default cache size if you choose. See “[UltraLite HotSync Conduit Installation utility for Palm OS \(ulcond10\)](#)” [*UltraLite Database User’s Guide*, page 142]

- ◆ **Predicates on publications** Synchronization publications for UltraLite now allow predicates. If you want to optionally combine conditional expressions with the logical operators AND and OR, you can now define this set of conditions in a WHERE or HAVING clause. As with SQL Anywhere, a predicate that evaluates to UNKNOWN is interpreted as FALSE. “[CREATE PUBLICATION statement](#)” [*UltraLite Database User’s Guide*, page 292] and “[ALTER PUBLICATION statement](#)” [*UltraLite Database User’s Guide*, page 288].

- ◆ **Improved MobiLink client network layer** Improvements to the client network layer include the following:

- Synchronization compression is available for all protocols.
- Persistent connections, so you can synchronize multiple times on the same connection.
- Resumable connections, to avoid data loss when a synchronization is interrupted.
- Introduction of IPv6 support.
- Improved error detection and debugging.

For more information on using UltraLite as a client to MobiLink, see “[UltraLite Clients](#)” [*MobiLink Clients*, page 345].

- ◆ **Set table order for synchronization** Synchronization from UltraLite clients now includes the ability to specify table ordering to avoid referential integrity issues during table upload. If you want to specify table order for synchronization, use the table_order synchronization parameter. See either

[“[\[Error! missing destination\]](#)” [*UltraLite Database User’s Guide*, page ??]]
or any of the following:

- ◆ UltraLite for MobileVB: “[ULSyncParms class](#)” [*UltraLite for Crossfire User’s Guide*, page 135]
- ◆ UltraLite.NET: “[ULSyncParms class](#)” [*UltraLite.NET User’s Guide*, page 293]
- ◆ UltraLite C/C++:

[“[\[Error! missing destination\]](#)” [*UltraLite C/C++ User’s Guide*, page ??]]

- ◆ UltraLite for M-Business Anywhere: “Class SyncParms” [*UltraLite for M-Business Anywhere User's Guide*, page 109]
- ◆ UltraLite for embedded SQL: “ULGetSynchResult function” [*UltraLite C/C++ User's Guide*, page 293]

Programming interfaces

General improvements

- ◆ **Cursor updates** UltraLite applications now support the ability to modify data in the database while processing a cursor. As with SQL Anywhere databases, not all query result sets allow cursor updates and deletes. Ensure you understand the cases in which cursor updates are allowed and executed. See “Fetching data” [*UltraLite C/C++ User's Guide*, page 54].
- ◆ **Simplified connection strings** Because the default user ID of **DBA** and password of **sql** are always provided by UltraLite, you can now connect by specifying only the database in your connection string. Furthermore, most databases can be set with the DBF connection parameter. See “Connecting to an UltraLite Database” [*UltraLite Database User's Guide*, page 53].
- ◆ **Introduction of MLFileTransfer functions** Use the file transfer function to download a file with the MobiLink file transfer utility. The file to be downloaded can be specific to a Mobilink username or a default file. For example, an application may choose to download a pre-configured empty database file to replace the local database (at beginning of month or processing cycle). See “MobiLink file transfer utility [mlfiletransfer]” [*MobiLink Clients*, page 31].
 - ◆ UltraLite for C/C++: “MLFileTransfer function” [*UltraLite C/C++ User's Guide*, page 158]
 - ◆ UltraLite.NET: “ULFileTransfer class” [*UltraLite.NET User's Guide*, page 178] and “ULFileTransferProgressData class” [*UltraLite.NET User's Guide*, page 188]
 - ◆ UltraLite for MobileVB: [nothing to link to]
 - ◆ UltraLite for M-Business Anywhere: [nothing to link to]
- ◆ **Database creation** The UltraLite schema is now part of the database rather than in a separate .usm file. This means that applications can no longer create a new database in the same way as was supported in earlier versions.

See any of the following:

- ◆ UltraLite for C/C++: “ULCreateDatabase function” [*UltraLite C/C++ User's Guide*, page 162]

- ◆ UltraLite .NET: “ULDatabaseManager members” [*UltraLite.NET User’s Guide*, page 126]
- ◆ UltraLite for MobileVB: “ULDatabaseManager class” [*UltraLite for Crossfire User’s Guide*, page 103]
- ◆ UltraLite for M-Business Anywhere: “Method createDatabase” [*UltraLite for M-Business Anywhere User’s Guide*, page 73]

UltraLite C/C++

- ◆ **Support for Symbian OS** UltraLite C/C++ support is now provided for the Symbian OS platform, using Codewarrior or Carbide C++ development environment.
- ◆ **New functions** Various new functions have been added in this release. These functions include:
 - The GetPublicationMask function gets the publication mask for a given publication name. See “IsCaseSensitive Function” [*UltraLite C/C++ User’s Guide*, page 232].
 - You must now call the appropriate ULEnable*Synchronization function before synchronizing over a specific network protocol. See “ULEnableHttpSynchronization function” [*UltraLite C/C++ User’s Guide*, page 167], “ULEnableHttpsSynchronization function” [*UltraLite C/C++ User’s Guide*, page 168], “ULEnableTcpiSynchronization function” [*UltraLite C/C++ User’s Guide*, page 172], “ULEnableTlsSynchronization function” [*UltraLite C/C++ User’s Guide*, page 173], “ULEnableZlibSyncEncryption function” [*UltraLite C/C++ User’s Guide*, page 175], or “ULEnableEccSyncEncryption function” [*UltraLite C/C++ User’s Guide*, page 164].
- ◆ **Improved support for wide and narrow (ASCII) characters** Although UltraLite now has one database file format (narrow characters), applications can still use wide definitions of TCHAR. Wide characters are converted to their MBCS equivalent and vice versa as appropriate.
- ◆ **Enhanced functions changes** Enhancements to existing functions include:
 - If your application does not require SQL support, using the ULEnableDatabaseManagerNoSQL function instead of ULEnableDatabaseManager can significantly reduce the size of the application. See “ULEnableDatabaseManagerNoSQL” [*UltraLite C/C++ User’s Guide*, page 177].
 - The SetReadPosition function has been enhanced to take a second parameter, offset_in_chars, which indicates if the offset is in bytes or characters. [Second parameter not doc’d.] See “SetReadPosition Function” [*UltraLite C/C++ User’s Guide*, page 252].

- SetSynchInfo now performs an autocommit so all synchronization information is immediately saved.
- ULStoreDefragInit, ULStoreDefragFini and ULStoreDefragStep are no longer required. UltraLite now internally manages database store defragmentation.
- ULEnableUserAuthentication is always enabled. UltraLite now permits the definition of up to four user names that may connect to the database (default is user name “DBA” with password “sql”).
- MobiLink synchronization code must now invoke ULEnableTcpiSynchronization() before invoking InitSynchInfo. See [“InitSynchInfo Function” \[UltraLite C/C++ User's Guide, page 214\]](#).

UltraLite Embedded SQL UltraLite Embedded SQL is no longer a static API, and no longer requires a reference database. Instead, the SQL preprocessor requires only the source files. It generates functions that send SQL statements to UltraLite. Some SQL statements that were supported in previous releases are not supported by UltraLite SQL. Version 10 supports dynamic SQL statements which were not supported in previous releases.

- ◆ **New functions** Various new functions have been added in this release. These functions include:
 - The ULEnableZlibSyncCompression function enables zlib compression during synchronization. See [“ULEnableZlibSyncEncryption function” \[UltraLite C/C++ User's Guide, page 175\]](#) and [“MobiLink Client Network Protocol Options” \[MobiLink Clients, page 33\]](#).

Notes

zlib compression is not supported for Palm OS or Symbian OS.

- ◆ **Enhanced functions** Enhancements to existing functions include:
 - GetSQLColumnName was added to UltraLite_RowSchema_iface. Depending on the type of schema, the function returns different results:
 - ◆ When used with a UltraLite_TableSchema, this function returns the column name specified by the column_id parameter.
 - When used with a UltraLite_ResultSetSchema, this function returns either:
 - An alias name, if one is specified for the result set column in question.
 - The column name, if the result set column represents a column in a table.
 - An empty string in all other cases.

UltraLite.NET

UltraLite now supports ADO.NET 1.0 development in Visual Studio 2003 and ADO.NET 2.0 development in Visual Studio 2005.

- ◆ **New methods** Various new functions have been added in this release. These functions include:
 - The `ExecuteResultSet` method executes a SQL `SELECT` statement and returns an updatable result set as a `ULResultSet` class. See [“ExecuteResultSet method”](#) [*UltraLite.NET User’s Guide*, page 60].
 - The `ULResultSet` class includes the following methods: `Append*`, `Set*`, `Delete`, `Update`. See [“ULResultSet class”](#) [*UltraLite.NET User’s Guide*, page 229] for details on these methods.
 - UltraLite.NET now supports TLS during TCP/IP synchronization. See [“ULStreamType enumeration”](#) [*UltraLite.NET User’s Guide*, page 292].
 - `ConnectionString` properties and the `ULConnectionParms` object have been enhanced to support limited quoting. See [“ULConnectionParms class”](#) [*UltraLite.NET User’s Guide*, page 91].
 - The `GetPublicationPredicate` method returns publication predicate string for the specified publication. If the publication does not exist, `SQL_E_PUBLICATION_NOT_FOUND` is set. See [“GetPublicationPredicate method”](#) [*UltraLite.NET User’s Guide*, page 344].
 - The `SignalSyncIsComplete` method signals the MobiLink provider for ActiveSync that an application has completed synchronization. See [“SignalSyncIsComplete method”](#) [*UltraLite.NET User’s Guide*, page 131].
 - The `SetDatabaseOption` method sets the value for the specified database option. See [“SetDatabaseOption method”](#) [*UltraLite.NET User’s Guide*, page 138].
- ◆ **Enhanced methods** Enhancements to existing methods include:
 - The `ULSyncParms` class now take a `TableOrder` order property to specify the order in which tables should be uploaded to the consolidated database. See [“TableOrder property”](#) [*UltraLite.NET User’s Guide*, page 301].
 - The `GetSchemaTable` method has now returns extended Table metadata. See [“GetSchemaTable method”](#) [*UltraLite.NET User’s Guide*, page 162] for a complete list.
 - The `UpdateBegin` method is now an optional at the `ResultSet` level when a table is in `UL_TABLE_ACCESS_MODE_NONE` or `UL_TABLE_ACCESS_MODE_FIND_AGAIN`. This change was required to make the UltraLite.NET API compatible with the ADO.NET 2.0 result set. See [“UpdateBegin method”](#) [*UltraLite.NET User’s Guide*, page 337].

UltraLite for AppForge
Crossfire

- The GetDatabaseProperty method now recognizes more properties. See “GetDatabaseProperty method” [*UltraLite.NET User's Guide*, page 135].
- The ULSyncProgressData class now includes a Flags property. See “Flags property” [*UltraLite.NET User's Guide*, page 306].

UltraLite for AppForge now supports the Symbian OS platform. Support for the UltraLite engine has been added in this release, allowing multiple applications to concurrently access a single database.

- ◆ **New methods** A new method has been added in this release. These functions include:
 - The OnWaiting method provides a mechanism for the user application to process GUI events and possibly cancel the current operation. In AppForge, you can use this method as part of the See [XX].

[(no reference topic exists. should it?)]

- ◆ **Enhanced methods** [So far none listed.]

- ◆ **New methods** A new method has been added in this release. These functions include:
 - The OnWaiting method provides a mechanism for the user application to process GUI events and possibly cancel the current operation. See [XX].

[(no reference topic exists. should it? should be part of ULConnection object.)]

- ◆ **Enhanced methods** Enhancements to existing methods include:
 - [Anything?]

UltraLite for M-Business
Anywhere

- ◆ **New methods** Various new methods have been added in this release. These functions include:
 - The setMBAServerWithMoreParms method sets proxy server information when using one-button synchronization. This new method enhances the existing setMBAServer method, by adding a new string argument named **additional**. [(no reference topic exists. should it?)] See [XX].
 - The getPublicationMask method gets the publication mask for a given publication name. See “Method getPublicationMask” [*UltraLite for M-Business Anywhere User's Guide*, page 110].
 - The getPublicationPredicate method returns publication predicate string for the specified publication. If the publication does not exist, `SQLC_PUBLICATION_NOT_FOUND` is set. [(no reference topic exists. should it?)] See [XX].

- ◆ **Enhanced methods** Enhancements to the following existing method includes:
 - The setStream method now supports ECC (Elliptic curve cryptography) for TLS (Transport Layer security). See “[Class SyncParms](#)” [*UltraLite for M-Business Anywhere User's Guide*, page 109].
[Warning: Constant topic does not have TLS listed. should it?]

Behavior changes and deprecated features

Following is a list of changes to UltraLite introduced in version 10.0.0.

Deprecated platforms

- ◆ Support for the PocketPC 2000 OS has been deprecated for this release.
- ◆ Support for CodeWarrior 8 has been removed. You must use Code Warrior 9 instead.

☞ For a list of supported platforms in UltraLite, see “[SQL Anywhere supported platforms](#)” [*Introducing SQL Anywhere 10*, page 19].

Removed components, modules, namespaces

The following programming interfaces have been dropped from this release:

- ◆ **UltraLite for ActiveX** All applications must be rewritten using a supported API.
- ◆ **Static Java API** All applications must be rewritten using a supported API.
- ◆ **Native UltraLite for Java** All applications must be rewritten using a supported API.
- ◆ **Static C++ API and Static embedded SQL** Developers wanting to write C++ applications must program using the dynamic C++ interface. If you have an application written with the static C++ library from previous versions, UltraLite 10 includes a migration utility to simplify the move to this new library. See “[Upgrading UltraLite](#)” on page 349.
- ◆ **iAnywhere.UltraLite namespace** In UltraLite.NET, this namespace is no longer supported. You must re-write your applications using the iAnywhere.Data.UltraLite namespace instead.

Removed utilities

- ◆ **Schema Painter** Because you no longer need a schema file to create an UltraLite database, the Schema Painter tool has been removed.

Removed, deprecated,
and modified functions

- ◆ **Database conversion tool** The Database conversion tool (the ulconv utility) is no longer supported. For the ulconv functionality, use the ulcreate, ulload, ulsync, and ulunload utilities.
- ◆ **ulxml utility** The ulxml utility that converted schema files to XML is no longer supported. For similar ulxml functionality, use ulload and ulunload to convert databases to XML instead.
- ◆ **ulisql** The ulisql utility is no longer supported. Instead, Interactive SQL (dbisql) now supports UltraLite.
- ◆ **ulgen** The ulgen utility is no longer supported. For UltraLite deployments that used this utility, you need to upgrade your database and C/C++ applications accordingly. See “[Upgrading UltraLite](#)” on page 349.
- ◆ **UltraLite for C/C++ API** Changes to functions and macros in the C/C++ API include:
 - The database schema can no longer be connected to nor upgraded dynamically because the .usm file no longer exists. All classes and/or functions relating to this former feature of UltraLite have been removed.
 - ULEnablePalmRecord and ULEnableFileDB have been removed in this version. [Recommendations?]
 - The macro UL_STORE_PARMS has been deprecated in release 10. Connection and creation options are specified in the appropriate parameter when calling OpenConnection or CreateDatabase.
 - ULSecureCerticomTLSStream and ULSecureRSATLSStream are deprecated in this release. In their place, you can use ULEccTlsStream and ULRsaTlsStream.
 - The security and security_parms fields of ul_synch_info are removed. Instead, set the stream field to the appropriate string value: tcpip, http, https or tls. Additionally, combine the security parameters with the other stream parameters. TCP/IP is always the underlying transport mechanism and TLS over HTTP is no longer supported. Instead you can use the HTTPS synchronization stream. See [“[Error! missing destination]” [UltraLite C/C++ User’s Guide, page ??]].
 - ULSocketStream, ULHTTPStream and ULHTTPSSStream have been changed to return the appropriate string value that is now required.
 - ULActiveSyncStream is removed from UltraLite. When synchronizing via ActiveSync an application should specify the stream value as with any other type of synchronization.
- ◆ **Embedded SQL** Changes to functions in the embedded SQL interface to the C/C++ API include:

- The database schema can no longer be upgraded dynamically because the *.usm* file no longer exists. All classes and/or functions relating to this former feature of UltraLite have been removed.
- ◆ **UltraLite.NET API** Changes to functions in the UltraLite.NET API include:
 - The database schema can no longer be connected to nor upgraded dynamically because the *.usm* file no longer exists. All classes and/or methods relating to this former feature of UltraLite have been removed.
 - ParmsUsed property has been renamed ToString in the ULConnectionParms class.
 - GetSQLColumnName has been renamed to GetColumnSQLName.
 - ULStreamType members UNKNOWN and ACTIVE_SYNC are removed from this enumeration. The default is now ULStreamType.TCPIP.
- ◆ **UltraLite for MobileVB API** Changes methods in the MobileVB API include:
 - The database schema can no longer be connected to nor upgraded dynamically because the *.usm* file no longer exists. All classes and/or methods relating to this former feature of UltraLite have been removed.
- ◆ **UltraLite for M-Business Anywhere API** Changes to functions in the M-Business Anywhere API include:
 - The database schema can no longer be connected to nor upgraded dynamically because the *USM* file no longer exists. All classes and/or methods relating to this former feature of UltraLite have been removed.
- ◆ **ulcond.log** Version 10 of the UltraLite HotSync conduit installer (ulcond10) no longer writes messages to this log file. For updated ulcond10 utility usage, see [“UltraLite HotSync Conduit Installation utility for Palm OS \(ulcond10\)”](#) [*UltraLite Database User’s Guide*, page 142].

Miscellaneous

Sybase Central and Interactive SQL

New features

Following is a list of additions to Sybase Central and Interactive SQL introduced in version 10.0.0.

Sybase Central

This section describes new features in Sybase Central. Changes and additions made to the Sybase Central plug-ins are described in the following sections:

- ◆ [“SQL Anywhere plug-in” on page 113](#)
- ◆ [“New plug-ins to Sybase Central” on page 114](#)
- ◆ **Sybase Central Task list** You can choose to view tasks in the left pane of Sybase Central, instead of a tree structure of the database. The Task list shows common tasks related to the object that is currently selected. The Task list includes common tasks, navigation options, and links to the documentation.
- ◆ **New Connections menu** In previous releases, you could press F11 to open the New Connection dialog that let you choose the plug-in you wanted to connect with. Sybase Central now provides a Connections menu where you can choose the plug-in you want to use.
- ◆ **Connection profile enhancements** You can now add descriptions to Sybase Central connection profiles. Connection profiles can also be imported and exported.
- ◆ **Plug-in searches** You can now search for objects that contain specified text within the plug-ins and databases in Sybase Central by choosing View ► Search Pane.
- ◆ **Context dropdown list** The new Context dropdown list shows you the object that is currently selected in the object tree to make it easier to navigate through plug-ins, especially when the object tree is not open in the left pane.

SQL Anywhere plug-in

- ◆ **Deadlocks database tab** When you are connected to a SQL Anywhere database in Sybase Central, you can view information about deadlocks on the Deadlocks tab. See [“Viewing deadlocks from Sybase Central” \[SQL Anywhere SQL User's Guide, page 146\]](#).
- ◆ **Entity-relationship diagrams** Sybase Central displays entity-relationship diagrams for databases, showing the tables in the database and their foreign key relationships. See [“Viewing entity-relationship diagrams in Sybase Central” \[SQL Anywhere Database Administration Guide, page 513\]](#).

- ◆ **New Schedule Creation wizard and other enhancements to the Events folder** For each scheduled event, the Events folder now displays the next scheduled time when the event will be triggered. For each conditional event, the folder displays the system event and optionally the trigger conditions that trigger the event. Schedules are created using the new Schedule Creation wizard. When creating a new scheduled event, the Event wizard allows you to create a single schedule, but you can later add additional schedules to an event if required. See [“Defining schedules” \[SQL Anywhere Database Administration Guide, page 776\]](#).
- ◆ **Maintenance plans** You can set up a schedule for validating and backing up a database automatically and have the output log emailed to you. See [“Creating a maintenance plan” \[SQL Anywhere Database Administration Guide, page 769\]](#).
- ◆ **New Restore Database wizard** [under construction]
- ◆ **Editor enhancements for Sybase Central and Interactive SQL** You can choose the font used in the editor windows of Sybase Central and Interactive SQL on the Format tab of the Options dialog.
A typing completion option has been added to the editor for database object names. See [“Using text completion” \[SQL Anywhere Database Administration Guide, page 537\]](#).
- ◆ **QAnywhere plug-in** The QAnywhere plug-in provides an easy-to-use graphical interface for creating and administering your QAnywhere applications.
☞ See [“New QAnywhere plug-in for Sybase Central” on page 91](#).
- ◆ **UltraLite plug-in** The UltraLite plug-in allows you to create, modify, and administer your UltraLite databases in a graphical user interface.
☞ See [“Graphical administration tools” on page 100](#).
- ◆ **MobiLink Create Synchronization Model wizard and Model mode** You can now create a synchronization model using a wizard, and edit your model in the MobiLink plug-in using the new Model mode. You can also set up MobiLink server-initiated synchronization. The old features of the MobiLink plug-in have also been enhanced and are preserved in Admin mode.
☞ See [“Enhancements to the MobiLink plug-in in Sybase Central” on page 68](#).

New plug-ins to Sybase Central

Interactive SQL

- ◆ **Interactive SQL can connect to UltraLite databases** You can now use Interactive SQL to develop and test SQL statements with UltraLite databases. The `ulsql` utility has been deprecated. See [“Graphical administration tools” on page 100](#).
- ◆ **Interactive SQL integrates with third party source control systems** Interactive SQL can integrate with third party source control systems, allowing you to perform a number of common source control operations on files from within Interactive SQL, such as checking in, checking out, and comparing against old versions. See [“Source control integration” \[SQL Anywhere Database Administration Guide, page 529\]](#).
- ◆ **New Interactive SQL options** The `isql_maximum_displayed_rows` option lets you specify the number of rows that appear in the result set in Interactive SQL, while the `isql_show_multiple_result_sets` option specifies whether multiple result sets can appear in the Results pane in Interactive SQL. See [“`isql_maximum_displayed_rows` option \[Interactive SQL\]” \[SQL Anywhere Database Administration Guide, page 404\]](#) and [“`isql_show_multiple_result_sets` \[Interactive SQL\]” \[SQL Anywhere Database Administration Guide, page 440\]](#).
- ◆ **Text completion** Interactive SQL now includes a typing completion option that can fill in the names of the following object types: tables, views, columns, stored procedures, and system functions. See [“Using text completion” \[SQL Anywhere Database Administration Guide, page 537\]](#).
- ◆ **DESCRIBE statement now supported by Interactive SQL** The DESCRIBE statement enables you to obtain the following information about a specified table or procedure:
 - ◆ all columns found in the table
 - ◆ all indexes found in the table
 - ◆ all parameters used with the stored procedure
 See [“DESCRIBE statement \[Interactive SQL\]” \[SQL Anywhere Reference, page 470\]](#).
- ◆ **Interactive SQL supports the @data option** When starting Interactive SQL from a command prompt, you can specify the `@data` option to read in options from the specified environment variable or configuration file. See [“Interactive SQL utility \(dbisql\)” \[SQL Anywhere Database Administration Guide, page 582\]](#).

SQL Anywhere Console
utility

- ◆ **SQL Anywhere Console supports the @data option** When starting the SQL Anywhere Console from a command prompt, you can specify

the `@data` option to read in options from the specified environment variable or configuration file. See “[SQL Anywhere Console utility \(dbconsole\)](#)” [*SQL Anywhere Database Administration Guide*, page 628].

Behavior changes and deprecated features

Following is a list of changes to Sybase Central and Interactive SQL introduced in version 10.0.0.

- ◆ **Interactive SQL no longer sets the `quoted_identifier` option to On**
In previous versions of the software, Interactive SQL set the `quoted_identifier` option to On. It now uses the database’s setting for this option (by default, this option is set to On). See “[quoted_identifier option \[compatibility\]](#)” [*SQL Anywhere Database Administration Guide*, page 434].
- ◆ **isql_plan option no longer supports the NONE parameter** The NONE parameter is no longer supported by the `isql_plan` option. See “[isql_plan option \[Interactive SQL\]](#)” [*SQL Anywhere Database Administration Guide*, page 405].
- ◆ **Interactive SQL can return unlimited result sets** In previous versions of the software, if you executed a query that returned multiple result sets, Interactive SQL only displayed a maximum of 10 result sets. Now Interactive SQL displays all result sets returned by the query. See “[isql_show_multiple_result_sets \[Interactive SQL\]](#)” [*SQL Anywhere Database Administration Guide*, page 440].
- ◆ **Interactive -f option behavior change** When starting Interactive SQL with the `-f` option, a connection is not made to a database automatically. Previously, a connection was opened automatically.
- ◆ **Accessing and saving graphical plans in Interactive SQL**
 - Two new menu choices, Open Plan and Save Plan, are available from the File menu in Interactive SQL for opening and saving graphical plans (previously this was done using the same Open and Save menu items as you used for opening and saving the SQL statements).
 - Previously, graphical plans were saved with an `.xml` file extension. Now, they are saved with the extension `.saplan`. However, the previous `.xml` file extension is still supported for displaying graphical plans that were stored with that extension.
 - In the Options dialog, you can now make Interactive SQL the default editor for both `.sql` and `.saplan` (graphical plan) files.

Deprecated and
discontinued features

- ◆ **SET OPTION statement TEMPORARY keyword deprecated** The TEMPORARY keyword is deprecated for setting Interactive SQL options using the SET OPTION statement. All Interactive SQL options set with SET OPTION are set permanently. See “[Interactive SQL options](#)” [*SQL Anywhere Database Administration Guide*, page 368].
- ◆ **EXIT statement now closes the current Interactive SQL window** In previous releases, executing an EXIT statement from Interactive SQL closed all the Interactive SQL windows. Now, only window where the statement is executed is closed. See “[EXIT statement \[Interactive SQL\]](#)” [*SQL Anywhere Reference*, page 499].
- ◆ **New error code for SQLE_ENGINE_NOT_MULTUSER** Applications that were programmed to handle SQLE_ENGINE_NOT_MULTUSER now need to check for a new error code. Previously, if an application attempted a write operation on the database while another thread was sending an upload to MobiLink, the runtime would return SQLE_ENGINE_NOT_MULTUSER. Now the runtime will return a new, more accurate error code: SQLE_ULTRALITE_WRITE_ACCESS_DENIED. See “[ULSQLCode enumeration](#)” [*UltraLite.NET User's Guide*, page 263].
- ◆ **jConnect no longer supported for connecting to Sybase Central, Interactive SQL, or the SQL Anywhere Console** Sybase Central, Interactive SQL, and the SQL Anywhere Console utility (dbconsole) no longer support connections to SQL Anywhere databases using jConnect. You can still use the iAnywhere JDBC driver to connect to databases from these applications. As a result of this change, the following features have been removed:
 - The -jconnect and -odbc options for the Interactive SQL utility (dbisql) have been removed.
 - The -jconnect and -odbc options for the SQL Anywhere Console utility (dbconsole) have been removed.
 - The Connect dialog used for connecting to Interactive SQL, the SQL Anywhere Console, and the SQL Anywhere and MobiLink plug-ins in Sybase Central no longer allows you to specify whether jConnect or the iAnywhere JDBC driver is used. The iAnywhere JDBC driver is used for all connections.
- ◆ **UltraLite plan now appears on the Plan tab in Interactive SQL** In previous releases, the Interactive SQL Results pane had an UltraLite Plan tab that displayed the UltraLite plan optimization strategy. The UltraLite Plan tab has been removed, so now when you are connected to an UltraLite database from Interactive SQL, the plan appears on the Plan tab.

- ◆ **Sybase Central SQL Anywhere plug-in no longer supports version 7 databases** Support for version 7 database servers and databases created with version 7 software has been removed from the SQL Anywhere plug-in. You can still connect to a database created with version 5, 6, or 7 software running on a version 8 or later database server for the purposes of unloading and reloading the database into a reload file, or into a new or existing database. See [“Rebuilding your database for version 10.0.0” on page 336](#).
- ◆ **isql_log option deprecated** The isql_log option for logging statements executed during an Interactive SQL session is deprecated. Use the START LOGGING and STOP LOGGING statements instead. See [“Logging commands” \[SQL Anywhere Database Administration Guide, page 522\]](#).
- ◆ **Sybase Central file name changes** In addition to file changes mentioned for Sybase Central, the following files have been added:



New name	Old name
asaplugin.jar	saplugin.jar

In addition to the changes mentioned for Sybase Central, the following registry key has changed

Old registry key	New registry key
HKLM\SOFTWARE\Sybase\Sybase Central\4.3	HKLM\SOFTWARE\Sybase\Sybase Central\5.0 (registry entries)

Documentation enhancements


The documentation for pre-existing features has been enhanced in several areas, including the following:

- ◆ **Context-sensitive help for SQL statements** In Interactive SQL, you can now right-click a SQL statement name to open the reference topic for the statement.
- ◆ **Getting Started with MobiLink** A new introductory book has been added that helps new users understand how to develop distributed applications with MobiLink. This book includes several new tutorials and new usage information.
 See [“Getting Started with MobiLink”](#) [*Getting Started with MobiLink*, page i].
- ◆ **Book removed** The book *ODBC Drivers for MobiLink* has been removed. This book is no longer necessary because of improvements to the ODBC drivers used by MobiLink.
- ◆ **New chapter about SQL Anywhere administration tools in Database Administration Guide** A new chapter has been added that focuses on how to use Sybase Central, Interactive SQL, and the SQL Anywhere Console utility.
 See [“SQL Anywhere Administration Tools”](#) [*SQL Anywhere Database Administration Guide*, page 509].
- ◆ **SQL Anywhere security features moved** The book *SQL Anywhere Studio Security Guide* has been removed. SQL Anywhere security features are now documented in *SQL Anywhere Database Administration Guide*, and MobiLink security features are documented in *MobiLink Administration Guide*.

Product-wide features

New features

Following is a list of product-wide additions introduced in version 10.0.0.

- ◆ **Product name changed to SQL Anywhere 10** SQL Anywhere Studio has been renamed SQL Anywhere 10, and Adaptive Server Anywhere has been renamed SQL Anywhere. Many names of files, directories, services, and executables have consequently changed, mostly to reflect the change from ASA to SA. These changes are detailed in the relevant Behavior Change topics of this chapter.
- ◆ **New install for DataWindow.NET** DataWindows.NET is a custom control tool that is useful to database developers using Visual Studio. It is provided as an optional component during your SQL Anywhere installation. Complete documentation is provided in the installation.
- ◆ **RSA now included with SQL Anywhere** You no longer have to purchase a separate license to use RSA encryption.
 See “MobiLink Transport-Layer Security” [[MobiLink Administration Guide](#), page 201].
- ◆ **User statistics collection** [under construction]
- ◆ **Error reporting** When an internal error occurs in Sybase Central, Interactive SQL, or the SQL Anywhere Console utility, a log of the error is written to the hard drive and a dialog appears where you can choose to send the error report to iAnywhere Solutions. The error log is deleted from your computer once you send the error report.
- ◆ **Control which features are available in the administration tools** You can now control which features are available for users in the administration tools using an *OEM.ini* file located in the same directory as tool's *.jar* file.

Behavior changes

Following is a list of product-wide changes in version 10.0.0.

- ◆ **Unsupported platforms** The following platforms are no longer supported:
 - ◆ Windows 95
 - ◆ Windows 98
 - ◆ Windows Me

- ◆ Windows NT
- ◆ Compaq Tru64

For other changes to platform support, see *components_platform_10.html*, located in the *support* subdirectory of your SQL anywhere installation.

- ◆ **Sample database revised and renamed** The SQL Anywhere sample database is now called *demo.db*. Objects such as tables, columns, views, and indexes now have full-word names to benefit users who use screen readers. See “SQL Anywhere sample database” [*Introducing SQL Anywhere 10*, page 63].
- ◆ **Miscellaneous file name changes** In addition to file name changes mentioned for individual products, the following file names have changed:

Old name	New name
<i>asa.cvf</i>	<i>sqlany.cvf</i>
<i>asaldap.ini</i>	<i>saldap.ini</i>
<i>asasrv.ini</i>	<i>sasrv.ini</i>
<i>asa_config.sh</i>	<i>sa_config.sh</i>
<i>install-dir/SYBSasa9/lib</i>	<i>sqlanywhere10/lib32</i> or <i>sqlanywhere10/lib64</i>
<i>install-dir/SYBSasa9/bin</i>	<i>sqlanywhere10/bin32</i> or <i>sqlanywhere10/bin64</i>

In addition to changes mentioned for individual products, the following registry keys have changed:

Old registry key	New registry key
<i>HKLM\SYSTEM\CurrentControlSet\Services\ASANYServer\9.0</i>	<i>HKLM\SYSTEM\CurrentControlSet\Services\ASANYServer\10.0</i>
<i>HKLM\SYSTEM\CurrentControlSet\Services\ASANYEngine\9.0</i>	<i>HKLM\SYSTEM\CurrentControlSet\Services\ASANYEngine\10.0</i>
<i>HKLM\SYSTEM\CurrentControlSet\Services\ASANYMLSync\9.0 Admin</i>	<i>HKLM\SYSTEM\CurrentControlSet\Services\ASANYMLSync\10.0 Admin</i>
<i>HKLM\SOFTWARE\Sybase\Sybase Central\4.3</i>	<i>HKLM\SOFTWARE\Sybase\Sybase Central\5.0 (registry entries)</i>
<i>HKLM\SOFTWARE\Sybase\Adaptive Server Anywhere\9.0</i>	<i>HKLM\SOFTWARE\Sybase\SQL Anywhere\10.0 (registry entries)</i>

In addition to file changes mentioned for individual products, the following file has been added:

New name
<i>ulbase.lib</i>

The following service group names have changed:

Description	Old name	New name
Network server	ASANYServer	SQLANYServer
Personal server	ASANYEngine	SQLANYEngine
MobiLink synchronization client	ASANYMLSync	SQLANYMLSync
Replication Agent	ASANYLTM	SQLANYLTM

Beta notes

This section lists limitations and unsupported features for the SQL Anywhere Beta.

You should also refer to the readme for the latest changes to the software. See *readme.txt*.

SQL Anywhere limitations

- ◆ 16 KB and 32 KB page sizes are not available.
Although you may find references to using these page sizes in the documentation, if you attempt to use an unsupported page size when creating or rebuilding a database, the operation will fail.
1 KB page sizes are no longer supported. You can still create databases with 2 KB, 4 KB, or 8 KB pages.
- ◆ The new SQL Flagger is not supported.
- ◆ The view definitions for some consolidated views refer to deprecated compatibility views. This will change so that all consolidated views will make use of system views and not deprecated compatibility views.
- ◆ The ECC and FIPS strong encryption libraries are not included. If you want to use strong encryption for transport-layer security or database encryption on Windows or Unix, you can use the AES and RSA strong encryption algorithms.

Query optimization limitations

- ◆ There is no low-memory strategy for hash joins. This affects the performance of RECURSIVE UNION queries.
- ◆ Parallel index scans are not supported.
- ◆ Partial index scans using a hash table scan from a hash join above an arbitrary subplan nested-block join (JNB) with sorted GROUPING SETS is not fully supported.
- ◆ The DISTINCT/GROUP BY index strategy is not supported.
- ◆ There is no alternative nested-loops strategy for hash joins.
- ◆ There is no efficient access plan for equality predicates over ROWID.
- ◆ The sort-based GROUPING SETS strategy is not supported.
- ◆ The clustered hash GROUP BY operator is not supported.
- ◆ The join nested block (JNB) is not supported.
- ◆ Low memory fallbacks are not supported.

MobiLink limitations

- ◆ Object-based data flow is not available in the .NET server API.
 - ◆ The ECC and FIPS strong encryption libraries are not included. If you want to use strong encryption for transport-layer security on Windows or Unix, you can use the AES and RSA strong encryption algorithms.
- UltraLite limitations
- ◆ The FIPS strong encryption libraries are not included. If you want to use strong encryption for database encryption, you can use obfuscation. For transport-layer security, only RSA is supported.
- Development and design tools limitations
- Data Window.NET, InfoMaker and PowerDesigner Physical Model are not available in Beta.

CHAPTER 2

What's New in Version 9.0.2

About this chapter

This chapter provides an overview of the new features and behavior changes introduced in SQL Anywhere Studio version 9.0.2.

Contents

Topic:	page
New features in version 9.0.2	126
Behavior changes in version 9.0.2	146

New features in version 9.0.2

This section lists the new features introduced in components of SQL Anywhere Studio version 9.0.2.

Adaptive Server Anywhere new features

This section introduces the new features in Adaptive Server Anywhere version 9.0.2. It provides an exhaustive listing of major and minor new features, with cross references to locations where each feature is discussed in detail.

SQL enhancements

- ◆ **UNIQUEIDENTIFIER native data type** The UNIQUEIDENTIFIER data type is now a native data type rather than a domain defined on BINARY(16). As a result, Adaptive Server Anywhere automatically carries out type conversions as needed, so that the STRTOUUID and UUIDTOSTR conversion functions are not needed to handle UNIQUEIDENTIFIER values.

To use the UNIQUEIDENTIFIER data type in databases created before this release, you must upgrade the database file format by unloading and reloading the database.

☞ For more information, see [“UNIQUEIDENTIFIER data type” \[SQL Anywhere Reference, page 76\]](#).
- ◆ **Conflict function for RESOLVE UPDATE triggers** The CONFLICT function can be used in conflict resolution triggers to determine if a particular column is a source of conflict for an UPDATE being performed on a SQL Remote consolidated database.

☞ For more information, see [“CONFLICT function \[Miscellaneous\]” \[SQL Anywhere Reference, page 118\]](#).
- ◆ **Procedure profiling enhancements** Profiling information can now be filtered per user and per connection using the sa_server_option stored procedure.

☞ For more information, see [“Enabling procedure profiling” \[SQL Anywhere SQL User’s Guide, page 255\]](#) and [“sa_server_option system procedure” \[SQL Anywhere Reference, page 907\]](#).
- ◆ **Remote servers can be tested before they are created or modified** The Remote Server Creation wizard in Sybase Central has a Test Connection button that allows you to test whether the connection information supplied in the remote server definition allows you to connect successfully before the remote server is created.

The Remote Server property sheet in Sybase Central also has a Test Connection button that allows you to test whether you can successfully connect to a remote server if its properties are changed.

☞ For more information, see [“Creating remote servers using Sybase Central”](#) [*SQL Anywhere SQL User's Guide*, page 709].

- ◆ **INPUT and OUTPUT statements accept the ESCAPES clause** The ESCAPES clause allows you to specify that characters are recognized and interpreted as special characters by the database server.

☞ For more information, see [“INPUT statement \[Interactive SQL\]”](#) [*SQL Anywhere Reference*, page 542] and [“OUTPUT statement \[Interactive SQL\]”](#) [*SQL Anywhere Reference*, page 577].

- ◆ **WAITFOR can wake up when it receives a message from another connection** The WAITFOR statement can now wake up when it receives a message from another connection using the MESSAGE statement.

☞ For more information, see [“WAITFOR statement”](#) [*SQL Anywhere Reference*, page 691].

- ◆ **Derived tables appear in query plans** Derived tables now appear as nodes in query execution plans.

☞ For more information, see [“Derived table”](#) [*SQL Anywhere SQL User's Guide*, page 530].

- ◆ **ALTER DOMAIN statement** The ALTER DOMAIN statement allows you to rename user-defined domains and data types.

☞ For more information, see [“ALTER DOMAIN statement”](#) [*SQL Anywhere Reference*, page 277].

- ◆ **NO RESULT SET clause for procedures** Declaring a stored procedure NO RESULT SET can be used when external environments need to know that the stored procedure does not return a result set.

☞ For more information, see [“CREATE PROCEDURE statement”](#) [*SQL Anywhere Reference*, page 391].

- ◆ **Column statistics updated during index creation** The CREATE INDEX statement now has the side effect that column statistics are updated for the indexed columns.

☞ For more information, see [“CREATE INDEX statement”](#) [*SQL Anywhere Reference*, page 382].

- ◆ **PHP module** The SQL Anywhere PHP module allows access to Adaptive Server Anywhere databases from the PHP scripting language.
☞ For more information, see “[SQL Anywhere PHP API](#)” [*SQL Anywhere Programming Guide*, page 471].
- ◆ **Web service clients** In addition to acting as a web-service provider, Adaptive Server Anywhere can now act as a web-service client, making it possible to create stored procedures and stored functions that access Adaptive Server Anywhere web services, as well as standard web services available over the internet.
☞ For more information, see “[SQL Anywhere Web Services Client Procedures and Functions](#)” [*SQL Anywhere Programming Guide*, page 551].
- ◆ **Multiple web service formats supported** The format of the WSDL file provided by a DISH service, as well as that of data payloads returned of part of SOAP responses, can now be selected to best suit the needs of the client applications. You can now choose between DNET for Microsoft .NET, CONCRETE for clients that automatically generate interfaces, and a general-purpose XML format.
For more information, see “[Creating SOAP and DISH web services](#)” [*SQL Anywhere Programming Guide*, page 529].
- ◆ **odbc_describe_binary_as_varbinary option** This option allows you to choose whether you want all BINARY and VARBINARY columns to be described to your application as BINARY or VARBINARY.
☞ For more information, see “[odbc_describe_binary_as_varbinary \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 420].
- ◆ **New prefetch option value** The prefetch option now has an additional value of Always. This value means that cursor results are prefetched even for SENSITIVE cursor types and cursors that involve a proxy table.
☞ For more information, see “[prefetch option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 431].
- ◆ **db_locate_servers_ex function** This function provides programmatic access to the information displayed by the dblocate -n option, listing all the Adaptive Server Anywhere database servers on a specific host.
☞ For more information, see “[db_locate_servers_ex function](#)” [*SQL Anywhere Programming Guide*, page 444].
- ◆ **SNMP Agent** Adaptive Server Anywhere can now be monitored from Simple Network Management Protocol (SNMP) applications.
☞ For more information, see “[The SQL Anywhere SNMP Extension Agent](#)” [*SQL Anywhere Database Administration Guide*, page 665].

Administrative
enhancements

- ◆ **Deadlock reporting** You can now obtain information about connections involved in deadlock using a new database option, `log_deadlocks`, and a new system stored procedure, `sa_report_deadlocks`. When you turn on the `log_deadlocks` option, the database server records information about deadlocks in an internal buffer. You can obtain deadlock information from this internal buffer by calling `sa_report_deadlocks`.

☞ For more information, see [“Determining who is blocked” \[SQL Anywhere SQL User's Guide, page 145\]](#).

- ◆ **New collations** The following collations have been added in this release:
 - **1252SWEFIN** has been added to support Swedish and Finnish. On Swedish and Finnish systems, the database server will choose 1252SWEFIN as the default collation for a new database if no collation is specified..
 - **1255HEB** has been added to support Hebrew. On Hebrew Windows systems, the database server will choose 1255HEB as the default collation for a new database if no collation is specified.
 - **1256ARA** has been added to support Arabic. On Arabic Windows systems, the database server will choose 1256ARA as the default collation for a new database if no collation is specified.
 - **950ZHO_HK and 950ZHO_TW** have been added to support Chinese. 950ZHO_HK provides support for the Windows Traditional Chinese character set cp950 plus the Hong Kong Supplementary Character Set (HKSCS). The 950ZHO_TW collation provides support for the Windows Traditional Chinese character set cp950, but doesn't support HKSCS. Ordering is based on a byte-by-byte ordering of the Traditional Chinese characters. These collations supercede the deprecated 950TWN collation.
 - **1252SPA** has been added to support Spanish. On Spanish Windows systems, the database server will choose 1252SPA as the default collation for a new database if a collation is not specified.
 - **874THAIBIN** has been added to support Thai. This is the recommended collation for Thai on both Windows and UNIX systems. For more information, see [“Supplied and recommended collations” \[SQL Anywhere Database Administration Guide, page 301\]](#).
- ◆ **New Service utility (dbsvc) options** The Service utility (dbsvc) supports the following new options:
 - **-cm option** This option displays the command used to create the specified service. This may be useful for deploying services, or for restoring them to their original state.

- **-sd option** This option allows you to provide a description of the service, which appears in the Windows Service Manager.
- **-sn option** This option allows you to provide a name for the service, which appears in the Windows Service Manager.

For more information, see “[Service utility options](#)” [*SQL Anywhere Database Administration Guide*, page 618].

- ◆ **New Data Source (dbdsn) utility options** The Data Source utility (dbdsn) supports the following new options:
 - **-cm option** This option displays the command used to create the specified data source. This may be useful for deploying data sources, or for restoring them to their original state.
 - **Driver connection parameter** You can use the Driver connection parameter to specify a driver for an ODBC data source when creating data sources using the Data Source utility (dbdsn) on Windows. On UNIX, if you do not specify the Driver connection parameter, the Data Source utility automatically adds a Driver entry with the full path of the Adaptive Server Anywhere ODBC driver based on the setting of the ASANY9 environment variable.

For more information, see “[Data Source utility options](#)” [*SQL Anywhere Database Administration Guide*, page 554].

- ◆ **Disk full callback support** The -fc database server option allows you to specify a DLL containing a callback function that can be used to notify users, and possibly take corrective action, when a file system full condition is encountered.

☞ For more information, see “[-fc server option](#)” [*SQL Anywhere Database Administration Guide*, page 145].

- ◆ **Validate Database wizard enhancements** When you validate a database using the Validate Database wizard in Sybase Central, the wizard indicates the current table being validated, as well as the overall progress of the validation operation. In addition, for databases with checksums enabled, you can validate both tables and checksums at the same time.

☞ For more information, see “[Validating a database](#)” [*SQL Anywhere Database Administration Guide*, page 747].

- ◆ **Unloading table data in Sybase Central** You can now unload data from one or more tables in Sybase Central in one step using the Unload Data dialog.

☞ For more information, see “[Using the Unload Data dialog](#)” [*SQL Anywhere SQL User's Guide*, page 678].

◆ **New columns added to `sa_index_density` and `sa_index_levels`**

Three new columns have been added to the result sets returned by the `sa_index_density` and `sa_index_levels` stored procedures: `TableId`, `IndexId`, and `IndexType`. If you want to revert to the old behavior of these stored procedures, you can drop the stored procedure and recreate it with the columns that were included in the result set in previous versions of the software.

☞ For more information, see “[sa_index_density system procedure](#)” [*SQL Anywhere Reference*, page 866] and “[sa_index_levels system procedure](#)” [*SQL Anywhere Reference*, page 869].

◆ **HISTORY option for BACKUP and RESTORE DATABASE statements**

The HISTORY option allows you to control whether BACKUP and RESTORE DATABASE operations are recorded in the `backup.syb` file.

☞ For more information, see “[BACKUP statement](#)” [*SQL Anywhere Reference*, page 318] and “[RESUME statement](#)” [*SQL Anywhere Reference*, page 608].

◆ **Support for integrated logins using Windows user groups** In addition to creating integrated logins for individual users on Windows NT/2000/XP, you can now create integrated login mappings to user groups on Windows NT/2000/XP. It is recommended that you upgrade your database before using this feature.

For more information, see “[Creating integrated logins for Windows user groups](#)” [*SQL Anywhere Database Administration Guide*, page 79].

◆ **Managing the size of the request log** The `-zn` database server option allows you to specify how many request log files should be retained.

☞ For more information, see “[-zn server option](#)” [*SQL Anywhere Database Administration Guide*, page 185].

◆ **Free pages at the end of the transaction log are removed when the file is renamed by a backup** Transaction log files are grown in fixed-size increments for better performance. When the transaction log is renamed as part of a backup, the free pages at the end of the log are removed, which helps free up disk space.

◆ **Remote server connections can now be explicitly closed** In previous releases, connections from Adaptive Server Anywhere to remote servers were disconnected only when a user disconnected from Adaptive Server Anywhere. You can now explicitly disconnect Adaptive Server Anywhere from a remote server using the new `CONNECTION CLOSE` clause of the `ALTER SERVER` statement.

☞ For more information, see “[ALTER SERVER statement](#)” [*SQL Anywhere Reference*, page 291].

Security enhancements

- ◆ **Initialization files can be obfuscated with dbfhide** The File Hiding utility (dbfhide) can now be used to obfuscate the contents of .ini files used by Adaptive Server Anywhere and its utilities.

☞ For more information, see “[File Hiding utility \(dbfhide\)](#)” [*SQL Anywhere Database Administration Guide*, page 566].

- ◆ **FIPS-certified security** On all supported Windows platforms except Windows CE, you can now use secure communication with FIPS 140-2 certified software from Certicom.

☞ For more information, see “[Starting the database server with transport-layer security](#)” [*SQL Anywhere Database Administration Guide*, page 854].

Strong database encryption using FIPS140-2 certified software from Certicom is also available on supported 32-bit Windows platforms.

☞ For more information, see “[Encrypting a database](#)” [*SQL Anywhere Database Administration Guide*, page 825].

Miscellaneous enhancements

- ◆ **New connection properties** The following connection properties have been added:

- ClientPort
- LoginTime
- ServerPort

For more information, see “[Connection-level properties](#)” [*SQL Anywhere Database Administration Guide*, page 458].

- ◆ **Proper formatting Event Viewer messages** When deploying Adaptive Server Anywhere databases, you should set a registry entry that controls the formatting of messages in the event viewer.

☞ For more information, see “[Deploying database servers](#)” [*SQL Anywhere Programming Guide*, page 690].

- ◆ **log_deadlocks option** This option allows you to control whether the database server logs information about deadlocks in an internal buffer. This option can be used with the sa_report_deadlocks procedure to obtain information about deadlock.

☞ For more information, see “[log_deadlocks option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 408].

- ◆ **rollback_on_deadlock option** This option allows you to control whether a transaction is automatically rolled back if it encounters a deadlock.
 - ☞ For more information, see “[rollback_on_deadlock \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 438].
- ◆ **temp_space_limit_check option** This option allows you to control what happens when a connection requests more than its quota of temporary file space.
 - ☞ For more information, see “[temp_space_limit_check option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 444].
- ◆ **New system stored procedures** Several new system stored procedures have been added:
 - **sa_rowgenerator procedure** The sa_rowgenerator system procedure is provided as an alternative to the RowGenerator table for returning a result set with rows between a specified start and end value. You can use this procedure for such tasks as generating a result set with rows for every value in a range or generating test data for a known number of rows in a result set.
 - ☞ For more information, see “[sa_rowgenerator system procedure](#)” [*SQL Anywhere Reference*, page 904].
 - **sa_send_udp stored procedure** This procedure sends a UDP packet to the specified address and can be used with MobiLink server-initiated synchronization to wake up the Listener utility (dblsn.exe).
 - ☞ For more information, see “[sa_send_udp system procedure](#)” [*SQL Anywhere Reference*, page 906].
 - **sa_verify_password stored procedure** This procedure is used by the sp_password stored procedure to verify the current user's password.
 - ☞ For more information, see “[sa_verify_password system procedure](#)” [*SQL Anywhere Reference*, page 925].
 - ☞ For more information, see “[sa_verify_password system procedure](#)” [*SQL Anywhere Reference*, page 925].
- ◆ **Maximum cache size on Windows CE** In previous releases of SQL Anywhere Studio, the maximum cache size on Windows CE was 32 MB. This limit has been removed and the cache size is now limited by the amount of available memory on the device.
- ◆ **New database server options for UNIX** The following database server options have been added for UNIX:

- **-uc** starts the database server in console mode on UNIX.
☞ For more information, see “-uc server option” [*SQL Anywhere Database Administration Guide*, page 176].
- **-ui** attempts to display the Server Startup Options dialog and Server Messages window when you start a database server on Linux and Solaris with X Windows support. If the server cannot find a usable display, the server starts in console mode.
☞ For more information, see “-ui server option” [*SQL Anywhere Database Administration Guide*, page 177].
- **-ux** displays the Server Startup Options dialog and Server Messages window when you start a database server on Linux and Solaris with X Windows support.
☞ For more information, see “-ux server option” [*SQL Anywhere Database Administration Guide*, page 178].

MobiLink new features

Following is a list of changes and additions to the software introduced in version 9.0.2.

◆ New Redirectors

- There is a new native Redirector for Apache, available on Windows, Solaris and Linux.
☞ See “Apache Redirector” [*MobiLink Administration Guide*, page 196].
- There is now an M-Business Anywhere Redirector, available on Windows, Solaris and Linux.
☞ See “M-Business Anywhere Redirector” [*MobiLink Administration Guide*, page 198].
- The NSAPI Redirector is now available on Solaris. Previously it was only available on Windows.
☞ See “NSAPI Redirector for Netscape/Sun web servers” [*MobiLink Administration Guide*, page 189].

◆ Protocols can now be configured to ignore specified hosts

A new option, ignore, can be used to specify hosts that should be ignored by the MobiLink synchronization server when they connect.

☞ See **ignore** in “-x option” [*MobiLink Administration Guide*, page 88].

◆ Prevent clients from waiting to synchronize when the MobiLink server is busy

You can now prevent clients from waiting to synchronize when the server is busy.

☞ For more information, see **backlog** in “-x option” [*MobiLink Administration Guide*, page 88]

- ◆ **Version stored in the consolidated database** The SQL Anywhere Studio version and build numbers are now stored in the MobiLink system table `ml_property`. For these entries, the `component_name` is **ML**, the `property_set_name` is **server_info**, the `property_name` is **release_version**, and the `property_value` is of the form *version.build*; for example, 9.0.2.1234.

☞ For more information about the MobiLink system table, see “[ml_property](#)” [*MobiLink Administration Guide*, page 607].

- ◆ **MobiLink synchronization server supports the new uniqueidentifier data type** The UNIQUEIDENTIFIER data type is now a native data type rather than a domain defined on BINARY(16). As a result, MobiLink remote databases now automatically carry out type conversions as needed, so that the String to UUID and UUID to String conversion functions are not needed to handle UNIQUEIDENTIFIER values.

☞ For information about the mapping of this data type to the supported consolidated databases, see “[MobiLink Data Mappings Between Remote and Consolidated Databases](#)” [*MobiLink Administration Guide*, page 633].

Security enhancements

- ◆ **FIPS-certified security streams** On Windows devices, you can now use secure communication with FIPS 140-2 certified software from Certicom.

☞ For more information, see “[Starting the MobiLink synchronization server with transport-layer security](#)” [*MobiLink Administration Guide*, page 214].

- ◆ **Connection options now shown in output log** MobiLink now displays the connection string and options in the output log, with passwords replaced with asterisks.

- ◆ **Deprecated security features** See “[MobiLink behavior changes](#)” on page 148.

MobiLink client enhancements

- ◆ **New synchronization setup tool for UltraLite** The UltraLite Schema Painter can now generate MobiLink synchronization scripts, as well as database tables and triggers for Adaptive Server Anywhere consolidated databases.

- ◆ **Now easier to delete a remote database and recreate it** The first synchronization of an Adaptive Server Anywhere client subscription now always works.

See “[Progress offsets](#)” [*MobiLink Clients*, page 77].

- ◆ **New dbmlsync hook is called when connections to MobiLink fail** A new event hook has been added, `sp_hook_dbmlsync_connect_failed`, that

allows you to program ways to recover from failed synchronization connections.

☞ See “[sp_hook_dbmlsync_ml_connect_failed](#)” [*MobiLink Clients*, page 254].

◆ **Improved integration of MobiLink clients into HTTP infrastructure**

You can now synchronize using HTTP when a proxy and/or web server requires RFC 2617 Basic or Digest authentication.

☞ See:

- ◆ “[http_password](#)” [*MobiLink Clients*, page 47]
- ◆ “[http_userid](#)” [*MobiLink Clients*, page 50]
- ◆ “[http_proxy_password](#)” [*MobiLink Clients*, page 48]
- ◆ “[http_proxy_userid](#)” [*MobiLink Clients*, page 49]

In addition, two new client connection parameters allow you to specify custom HTTP headers and custom cookies. In order to respect session cookies, HTTP clients now recognize all Set-Cookie and Set-Cookie2 HTTP headers that they receive in server replies and will send these cookies back up with all future HTTP requests. If the name of a cookie matches an existing cookie, the client will replace its old value with the new one. Cookies are not remembered between synchronizations: they are discarded at the end of the synchronization.

☞ See “[custom_header](#)” [*MobiLink Clients*, page 43] and “[set_cookie](#)” [*MobiLink Clients*, page 57].

◆ **Assistance in detecting connection errors**

MobiLink clients now issue a warning message when invalid connection parameters are specified.

◆ **Mirror log location**

When dbmlsync is run on a different computer from the remote database, or when mirror logs are located in a different directory from mirror transaction logs, dbmlsync is now able to automatically delete old log files when you specify the location of old mirror logs using this new extended option.

See “[MirrorLogDirectory \(mld\) extended option](#)” [*MobiLink Clients*, page 146].

Server-initiated
synchronization
enhancements

◆ **Enhanced functionality for connection-initiated synchronization**

In addition to `_BEST_IP_CHANGED_`, Windows Listeners now also generate the internal message `_IP_CHANGED_` to help you initiate synchronization when there is a change in connectivity.

See “[Connection-initiated synchronization](#)” [*MobiLink Server-Initiated Synchronization User's Guide*, page 35].

- ◆ **Listener post action enhancements** When you specify Listener post actions, you can now optionally use a Windows message ID to specify the window message, and can optionally use the window title instead of the window class. You can also use single quotes around the window class name or message if your message or title include non-alphanumeric characters such as spaces or punctuation marks.

☞ See **post** in “[Listener syntax](#)” [*MobiLink Server-Initiated Synchronization User's Guide*, page 42].

- ◆ **New action variables** There are several new action variables:

- \$request_id
- \$best_ip
- \$best_adapter_name
- \$best_adapter_mac
- \$best_network_name

See “[Action variables](#)” [*MobiLink Server-Initiated Synchronization User's Guide*, page 50].

- ◆ **More device support** The Palm Listener now supports Kyocera 7135 and Treo 600 smartphones.

☞ See “[Listeners for Palm Devices](#)” [*MobiLink Server-Initiated Synchronization User's Guide*, page 55].

SQL Remote new features

Following is a list of changes and additions to the software introduced in version 9.0.2.

- ◆ **Mirror log location** When dbremote is run on a different computer from the remote database, or when mirror logs are located in a different directory from mirror transaction logs, dbremote can automatically delete old log files if you specify the location of old mirror logs using the new -ml option.

☞ For more information, see “[Message Agent](#)” [*SQL Remote User's Guide*, page 188].

- ◆ **Conflict function for RESOLVE UPDATE triggers** The CONFLICT function can be used in conflict resolution triggers to determine if a particular column is a source of conflict for an UPDATE being performed on a SQL Remote consolidated database.

☞ For more information, see “[CONFLICT function \[Miscellaneous\]](#)” [*SQL Anywhere Reference*, page 118].

UltraLite new features

Following is a list of changes and additions to the software introduced in version 9.0.2.

- Component new features
- ◆ **ADO.NET interface in UltraLite.NET** UltraLite.NET now supports the ADO.NET programming interface in the new namespace `iAnywhere.Data.UltraLite`. ADO.NET provides an industry-standard interface to UltraLite, and also provides an easy migration path to Adaptive Server Anywhere for large applications.

The ADO.NET interface is recommended over the previous UltraLite.NET interface (`iAnywhere.UltraLite` namespace), which is now deprecated.

☞ For more information, see “[Tutorial: Build an UltraLite.NET Application](#)” [*UltraLite.NET User’s Guide*, page 25] and “[iAnywhere.Data.UltraLite namespace](#)” [*UltraLite.NET User’s Guide*, page 41].
 - ◆ **UltraLite for MobileVB enhancements** UltraLite for MobileVB now supports Visual Basic .NET programming using AppForge Crossfire.

☞ For more information, see [UltraLite for Crossfire User’s Guide](#).
 - ◆ **UltraLite for M-Business Anywhere enhancements** The following enhancements have been made to UltraLite for M-Business Anywhere:
 - UltraLite for M-Business Anywhere now supports the client/server UltraLite engine. Your application can use the `DatabaseManager.runtimeType` property to inspect whether the engine or the runtime library is being used.
 - UltraLite for M-Business Anywhere applications can now synchronize both data and web content with a single operation.

☞ See “[One-button synchronization](#)” [*UltraLite for M-Business Anywhere User’s Guide*, page 31].
 - You can use a MobiLink Redirector to synchronize both data and web content through a single M-Business Anywhere server. For synchronization from outside firewalls, this reduces the number of ports that need to be accessible.

☞ See “[Synchronizing data via M-Business Anywhere](#)” [*UltraLite for M-Business Anywhere User’s Guide*, page 33] and “[M-Business Anywhere Redirector](#)” [*MobiLink Administration Guide*, page 198].
 - M-Business Anywhere 5.5 on Windows XP is now a supported platform. The connection parameters `databaseOnDesktop` and `schemaOnDesktop` support this environment.

☞ See “Class ConnectionParms” [*UltraLite for M-Business Anywhere User's Guide*, page 70].

- Additional methods have been added to the API that enable you to gather information about data using the column ID rather than the column name.

☞ See “Class ResultSetSchema” [*UltraLite for M-Business Anywhere User's Guide*, page 100] and “Class TableSchema” [*UltraLite for M-Business Anywhere User's Guide*, page 124].

For more information, see *UltraLite for M-Business Anywhere User's Guide*.

- ◆ **Native UltraLite for Java enhancements** The following enhancements have been made to Native UltraLite for Java:
 - Column schema info accessible by columnID instead of just name.
 - New SyncProgressData ErrorMessage property and improved sync error reporting.
 - PreparedStatement.[get]Plan added.
 - ResultSet, ResultSetSchema keep PreparedStatement alive while in use.
- ◆ **UltraLite.NET component enhancements** The following functions are supported by UltraLite.NET. It is recommended that these functions be used as part of the ADO.NET interface (iAnywhere.Data.UltraLite namespace).
 - New ULCursorSchema.Name, ULResultSetSchema.Name read-only properties.
 - New ULSyncProgressData ErrorMessage property and improved sync error reporting.
 - ULCommand.Plan read-only property.

For more information, see “iAnywhere.Data.UltraLite namespace” [*UltraLite.NET User's Guide*, page 41].
- ◆ **Palm developers can now use a version-independent prefix file** In previous releases, the UltraLite prefix file depended on the version of Palm OS for which you were developing. You can now use *ulpalmos.h* for any version of Palm OS.

☞ See “Using the UltraLite plug-in for CodeWarrior” [*UltraLite C/C++ User's Guide*, page 83].
- ◆ **Palm developers can now use expanded mode** CodeWarrior supports a code generation mode called **expanded mode**, which

improves memory use for global data. You can now use an expanded mode version of the UltraLite runtime library.

☞ For more information, see [“Building Expanded Mode applications” \[UltraLite C/C++ User’s Guide, page 84\]](#).

- ◆ **Trusted certificates can be retrieved from permanent storage** In previous releases of the software, the trusted certificate for secure synchronization was embedded in the database schema. On Windows and Windows CE platforms, it can now be stored externally and accessed via the `trusted_certificates` option.

☞ For more information, see [“trusted_certificates” \[MobiLink Clients, page 61\]](#).

SQL and runtime enhancements

- ◆ **Dynamic SQL enhancements** The following enhancements have been made to the UltraLite dynamic SQL support:

- **Query optimization improvement** In previous versions of the software, the order in which tables were accessed was the order in which they appeared in the query. In this version, UltraLite optimizes the query to find an efficient order in which to access tables. As long as you have defined appropriate indexes in the database, the optimizer helps to improve query execution performance.

- **Query plan viewing** Query access plans now include the index name instead of an index number, for clarity. Access plans can be seen, for example, from the new UltraLite Interactive SQL utility.

- **IF and CASE expressions** The range of expressions supported by UltraLite has been extended by adding the IF and CASE conditional expressions.

☞ For more information, see [“IF expressions” \[UltraLite Database User’s Guide, page 201\]](#) and [“CASE expressions” \[UltraLite Database User’s Guide, page 202\]](#).

- **Table names can have owner names** UltraLite tables do not have owners. Support has been added for `owner.table-name` as a convenience for existing SQL and for programmatically-generated SQL. UltraLite accepts but ignores `owner`.

- ◆ **UNIQUEIDENTIFIER data type introduced** The UNIQUEIDENTIFIER data type is now a native data type rather than a domain defined on BINARY(16). As a result, UltraLite automatically carries out type conversions as needed, so that the String to UUID and UUID to String conversion functions are not needed to handle UNIQUEIDENTIFIER values.

☞ For more information, see [“UNIQUEIDENTIFIER data type” \[SQL Anywhere Reference, page 76\]](#).

Administration
enhancements

- ◆ **UltraLite query plan descriptions enhanced** UltraLite query plan descriptions, which can be viewed in UltraLite Interactive SQL, have been enhanced to be easier to read for better diagnosis of performance issues.
- ◆ **UltraLite Interactive SQL utility** An UltraLite Interactive SQL utility is now provided for testing SQL statements against UltraLite databases and for modifying UltraLite data. It also displays query plans so that you can diagnose performance problems.
- ◆ **Command line utilities for database management** A set of command line utilities makes database management tasks easier for UltraLite files on Windows computers. These utilities are particularly useful during application development.

Each of the new utilities carries out a subset of the tasks that the ulconv utility provides. In future versions of the software, the ulconv utility will be replaced by these newer single-task utilities.

For more information, see:

- ◆ “UltraLite Database Creation utility (ulcreate)” [*UltraLite Database User's Guide*, page 146]
- ◆ “UltraLite Load XML to Database utility (ulload)” [*UltraLite Database User's Guide*, page 158]
- ◆ “UltraLite Synchronization utility (ulsync)” [*UltraLite Database User's Guide*, page 163]
- ◆ “UltraLite Unload Database to XML utility (ulunload)” [*UltraLite Database User's Guide*, page 166]

Synchronization
enhancements




- ◆ **Improved integration of MobiLink clients into HTTP infrastructure** Two new client connection parameters allow you to specify custom headers and custom cookies.
For more information, see “[custom_header](#)” [*MobiLink Clients*, page 43] and “[set_cookie](#)” [*MobiLink Clients*, page 57].
- ◆ **Synchronization script generation from the Schema Painter** The UltraLite Schema Painter now provides the ability to generate synchronization scripts for Adaptive Server Anywhere consolidated databases. This capability makes it easier to extend UltraLite applications to a synchronized architecture.
- ◆ **Synchronization notifications on referential integrity violations** Support synchronization callback functions to report referential integrity violations - currently, rows that fail RI are silently deleted.

QAnywhere new features

Transmission rule
enhancements

QAnywhere Agent
enhancements

- ◆ **Failover servers** The QAnywhere agent now can take a list of MobiLink server connection protocol options rather than just one.
☞ See “[Setting up a failover mechanism](#)” [*QAnywhere User’s Guide*, page 54].
- ◆ **Emulator support** QAnywhere client applications on the Pocket PC 2002 and Pocket PC 2003 now support x86 emulators. Only “scheduled” policy for the QAnywhere Agent is supported on these emulators.
- ◆ **New RDBMSs supported as server message stores** All supported MobiLink consolidated databases can now be used in QAnywhere applications as server message stores: Adaptive Server Anywhere, Adaptive Server Enterprise, Microsoft SQL Server, Oracle, and DB2.
- ◆ **QAnywhere .NET client library for the .NET Compact Framework now supports message listeners.** QAnywhere .NET client library for the .NET Compact Framework now supports message listeners.
- ◆ **Remote message store properties now synchronized** When you set remote message store properties, those properties are now synchronized to the server message store so that they can be used in transmission rules.
- ◆ **Enhanced message store properties** The `ias_Network` property now contains fields you can use to access detailed network information.
☞ See “[Message store properties](#)” [*QAnywhere User’s Guide*, page 179].
In addition, you can now create customized message store properties.
- ◆ **Rules for deleting messages** You can now specify transmission rules for the persistence of messages in the message stores. You can delete messages on the client side and server side.
☞ For more information, see “[Transmission rules](#)” [*QAnywhere User’s Guide*, page 188].
- ◆ **Connection string** To start the local message store, you can now specify a connection string with the `qaagent -c` option. This allows you to use Adaptive Server Anywhere connection string parameters.
☞ See “[-c option](#)” [*QAnywhere User’s Guide*, page 126].
- ◆ **Quiet mode** The QAnywhere Agent now supports two flavors of quiet mode, which can avoid problems on some Windows CE devices.
☞ See “[-q option](#)” [*QAnywhere User’s Guide*, page 143] and “[-qi option](#)” [*QAnywhere User’s Guide*, page 144].

- ◆ **QAstop utility** When you start the QAnywhere Agent in quiet mode with the `-qi` option, you must use the new `qastop` utility to stop it.
 See “`-qi` option” [*QAnywhere User's Guide*, page 144].
- ◆ **Enhanced verbosity** You can now specify output log file names with the `-o` or `-ot` option, and regulate the size of the output files with the `-os` and `-ot` options. In addition, the `-v` option replaces the old `-verbose` option. With `-v`, you have greater control over logging output.
 See:
 - ◆ “`-o` option” [*QAnywhere User's Guide*, page 134]
 - ◆ “`-ot` option” [*QAnywhere User's Guide*, page 137]
 - ◆ “`-on` option” [*QAnywhere User's Guide*, page 135]
 - ◆ “`-os` option” [*QAnywhere User's Guide*, page 136]
 - ◆ “`-v` option” [*QAnywhere User's Guide*, page 149]
- ◆ **Initialize database for use as a remote message store** You can use the new `qaagent -si` option to set up a remote message store. For more information, see “`-si` option” [*QAnywhere User's Guide*, page 145].
- ◆ **Upgrade from version 9.0.1** The QAnywhere Agent has a new option, `-su`, that upgrades a remote message store from version 9.0.1 to 9.0.2.
 See “`-su` option” [*QAnywhere User's Guide*, page 147].

QAnywhere MobiLink system tables

All QAnywhere MobiLink system tables are now owned by `ml_qa_user_group`. Previously, they were owned by `DBO`.

Two new MobiLink system tables have been added. For more information, see:

- ◆ “`ml_qa_delivery`” [*MobiLink Administration Guide*, page 609]
- ◆ “`ml_qa_delivery_client`” [*MobiLink Administration Guide*, page 610]

There are changes to the schema of several MobiLink system tables. For more information, see:

- ◆ “`ml_qa_global_props`” [*MobiLink Administration Guide*, page 611]
- ◆ “`ml_qa_global_props_client`” [*MobiLink Administration Guide*, page 612]
- ◆ “`ml_qa_repository`” [*MobiLink Administration Guide*, page 614]
- ◆ “`ml_qa_repository_client`” [*MobiLink Administration Guide*, page 615]
- ◆ “`ml_qa_repository_props`” [*MobiLink Administration Guide*, page 617]
- ◆ “`ml_qa_repository_client`” [*MobiLink Administration Guide*, page 615]

The following MobiLink system tables are not generated for 9.0.2 clients:

- ◆ ml_qa_repository_staging_client
- ◆ ml_qa_status_staging_client

The following MobiLink system table is not generated for 9.0.2 servers:

- ◆ ml_qa_repository_content

Documentation enhancements

This section introduces enhancements made to the appearance, organization, or navigation of the Adaptive Server Anywhere documentation for version 9.0.2. It provides an exhaustive listing of major changes.

New documentation

The documentation for existing features has been enhanced in several areas, including the following:

- ◆ **SNMP Agent documentation** A new book has been added that describes the Adaptive Server Anywhere SNMP Agent.
☞ See “[The SQL Anywhere SNMP Extension Agent](#)” [*SQL Anywhere Database Administration Guide*, page 665].
- ◆ **Windows CE starting points** A chapter containing starting points for Windows CE users has been added.
☞ See “[SQL Anywhere for Windows CE](#)” [*SQL Anywhere Database Administration Guide*, page 929].
- ◆ **DBTools interface to the MobiLink synchronization client** A sample and other information about how to use dbmlsync from DBTools has been added.
☞ See “[DBTools Interface for dbmlsync](#)” [*MobiLink Clients*, page 311].
- ◆ **QAnywhere enhancements** The QAnywhere documentation has been expanded, with new information about how to integrate messaging with JMS messaging systems and MobiLink data synchronization, and enhanced information about setting up QAnywhere applications.
☞ See [QAnywhere User's Guide](#).
- ◆ **Server-initiated synchronization SDKs** The documentation for the SDKs has been expanded, and a new section on the Palm Listener SDK has been added.
☞ See “[MobiLink Listener SDK for Palm](#)” [*MobiLink Server-Initiated Synchronization User's Guide*, page 95].

- ◆ **MobiLink reorganization** The MobiLink books have been reorganized so that there is now a client guide, an administration guide, and a book of tutorials. As well as covering Adaptive Server Anywhere clients, the client guide includes synchronization parameters and synchronization connection parameters for UltraLite clients, which were previously in the UltraLite Database User's Guide.
- ◆ **UltraLite API and QAnywhere API references** The UltraLite.NET, UltraLite C++ API, QAnywhere .NET, and QAnywhere C++ API material is now available in the same form as the remainder of the documentation. As a result, it is available as PDF as well as in the HTML-based documentation.

Behavior changes in version 9.0.2

This section lists the behavior changes introduced in components of SQL Anywhere Studio version 9.0.2. It also lists deprecated features, which are supported in the current software but will not be supported in the next major release of SQL Anywhere Studio.

Deprecated feature lists subject to change

As with all forward-looking statements, the lists of deprecated features are not guaranteed to be complete and are subject to change.

Adaptive Server Anywhere behavior changes

Deprecated and discontinued features

The following is a list of features that are no longer supported or are deprecated, and that may impact existing applications.

- ◆ **min_table_size_for_histogram option removed** The database server no longer uses the min_table_size_for_histogram option. In previous versions of the software, this option allowed you to specify the minimum table size for which histograms were created. Now Adaptive Server Anywhere automatically creates histograms for all tables with five or more rows. You can create histograms for all tables, regardless of size, using the CREATE STATISTICS statement.

☞ For more information, see [“Updating column statistics” \[SQL Anywhere SQL User’s Guide, page 508\]](#).

- ◆ **Deprecated database options** The following database options are no longer supported:
 - truncate_date_values
 - assume_distinct_servers
- ◆ **Old database formats deprecated** In the next major release of SQL Anywhere Studio, databases created under old versions of the software will not be supported. Migration tools will be provided.
- ◆ **Non-threaded DBTools library for UNIX deprecated** The non-threaded DBTools library for UNIX (*libdbtool9.so*) is deprecated: it is fully supported in the current software but will not be supported in the next major release of SQL Anywhere Studio.

- ◆ **950TWN collation no longer supported** The 950TWN has been superseded by the following collations: 950ZHO_HK and 950ZHO_TW.

☞ For more information, see [“Supplied and recommended collations” \[SQL Anywhere Database Administration Guide, page 301\]](#).

Other behavior changes The following is a list of behavior changes from previous versions of the software.

- ◆ **Restrictions on the Transaction Log utility (dblog) when removing the transaction log** When removing a transaction log using the `-n` option, you must also specify the corresponding ignore transaction log offset option (`-il` for the Log Transfer Manager, `-ir` for SQL Remote, or `-is` for `dbmlsync`).

☞ For information, see “[Transaction log utility options](#)” [*SQL Anywhere Database Administration Guide*, page 639].

- ◆ **Database utilities running in quiet mode** When performing any of the following operations with the `-q` option (quiet mode) specified, you must also specify the `-y` option:
 - modifying or deleting a service with the Service Creation (`dbsvc`) utility
 - modifying or deleting a datasource with the Data Source (`dbdsn`) utility
 - erasing a file with the Erase (`dberase`) utility
 - translating a transaction log with the Log Translation (`dbtran`) utility

- ◆ **Certificate name and password must be supplied when using ECC_TLS or RSA encryption** The default values for the `certificate`, `certificate_password`, and `trusted_certificates` parameters have been removed. These defaults utilized the sample certificates that are provided in the `win32` directory of your SQL Anywhere Studio installation. The sample certificates are useful only for testing and development purposes and do not provide security.

In addition, the `-ec all` server option is no longer supported.

For more information, see “[-ec server option](#)” [*SQL Anywhere Database Administration Guide*, page 141].

- ◆ **-xs server option change** The `-xs all` server option is no longer supported to listen for web requests on both HTTP and HTTPS ports.

☞ For more information, see “[-xs server option](#)” [*SQL Anywhere Database Administration Guide*, page 182].

- ◆ **TCP/IP port number must be specified for network database servers on Mac OS X, HP-UX, and Tru64 when the default port is not in use** If you are starting a database server on Mac OS X, HP-UX, or Tru64, you must specify a port number using the `ServerPort [PORT]` protocol option if the default port (2638) is already in use or if you do not want to use the default port.

☞ For more information, see “[ServerPort protocol option \[PORT\]](#)” [*SQL Anywhere Database Administration Guide*, page 247].

- ◆ **Dbospace file names are changed for databases unloaded and reloaded with the Unload utility (dbunload)** When a database is unloaded and reloaded using the -an option of the Unload utility (dbunload), the dbospace file names for the new database have an **R** appended to the end of the file name. This is done to prevent naming conflicts when the new dbospace files are placed in the same directory as the original dbospace files. Dbospace file names also have an R appended to the file name when you unload and reload data using the Unload Database wizard in Sybase Central.

☞ For more information, see “[Unload utility options](#)” [*SQL Anywhere Database Administration Guide*, page 648].

- ◆ **Property functions now return LONG VARCHAR values** Previously, the following functions returned a VARCHAR(254) value. They now return a VARCHAR(maxpropsize) value, where maxpropsize is based on the maximum page size specified for the server.
 - CONNECTION_PROPERTY
 - DB_EXTENDED_PROPERTY
 - DB_PROPERTY
 - EVENT_PARAMETER
 - PROPERTY
- ◆ **STRTOUUID function change** In previous releases, if STRTOUUID was passed an invalid UUID value it returned NULL. It now returns a conversion error unless the conversion_error option is set to OFF, in which case it returns NULL.

MobiLink behavior changes



The following is a list of behavior changes from previous versions of the software.

Security behavior changes

- ◆ **HTTP+TLS security deprecated in favor of HTTPS** Transport-layer security is deprecated for clients connecting over HTTP. To use transport-layer security over HTTP, you should use HTTPS.

☞ For information about server-side security, see “[Starting the MobiLink synchronization server with transport-layer security](#)” [*MobiLink Administration Guide*, page 214].

☞ For information about client-side security, see “[Configuring MobiLink clients to use transport-layer security](#)” [*MobiLink Administration Guide*, page 216].

- ◆ **Certificate name and password must be supplied when using ECC_TLS or RSA encryption with MobiLink** The default values for the certificate, certificate_password, and trusted_certificates synchronization parameters have been removed. These defaults utilized the sample certificates that are provided in the `win32` directory of your SQL Anywhere Studio installation. The sample certificates are useful only for testing and development purposes and do not provide security.
 For more information, see “-x option” [*MobiLink Administration Guide*, page 88].
- Other MobiLink behavior changes ◆ **No polling interval for UDP listening** On the Listener, there is now no polling interval for UDP connections. The Listener processes messages immediately.
 For more information, see **-I option** in “Listener Utility” [*MobiLink Server-Initiated Synchronization User's Guide*, page 41].
- ◆ **Support for MobiLink Palm Listener on Treo 180 and Kyocera 6035 smartphones deprecated** For information about supported devices for the Palm Listener, see “Listeners for Palm Devices” [*MobiLink Server-Initiated Synchronization User's Guide*, page 55].

SQL Remote behavior changes


Deprecated and discontinued features

The following is a list of features that are no longer supported or are deprecated, and that may impact existing applications.

- ◆ **SQL Remote for Adaptive Server Enterprise deprecated** In the next major release of SQL Anywhere Studio, SQL Remote for Adaptive Server Enterprise will not be present. MobiLink provides a more flexible and scalable solution for data synchronization between Adaptive Server Enterprise and Adaptive Server Anywhere databases.

Other behavior changes

The following is a list of behavior changes from previous versions of the software.

- ◆ **The Extraction (dbxtract) utility** When extracting a remote database with dbxtract, if the `-q` option (quiet mode) is specified, you should also specify the `-y` option so that dbxtract will automatically replace the existing command file without confirmation.
 For more information, see “Extraction utility” [*SQL Remote User's Guide*, page 198].
- ◆ **IPM_Receive message control parameter** The default value for the MAPI IPM_Receive message control parameter has been changed to

YES. Setting this value to YES ensures that both IPC and IPM messages are picked up by SQL Remote.

☞ For more information, see “The MAPI message system” [*SQL Remote User's Guide*, page 141].

UltraLite behavior changes

The next major release of UltraLite will enhance development using industry standard APIs and will enhance development using the component model as opposed to the original static interfaces. These changes will have several benefits for users, including making it easier to develop applications using UltraLite.

As a result of these plans, several UltraLite APIs are deprecated with this release, meaning that they continue to be fully supported in the current software but will not be supported in the next major release. Assistance in migrating applications that use deprecated interfaces will be provided in the next major release.

As with all forward-looking statements, the list of deprecated and discontinued features provided here is subject to change.

Deprecated and discontinued features

The following features are deprecated or discontinued.

- ◆ **Static interfaces deprecated** The next major release of SQL Anywhere Studio will not support the static C++ API or the static Java API. An embedded SQL interface will be available, but not through the current generated code mechanism.
- ◆ **UltraLite.NET component interface to be superseded by ADO.NET** In this release, UltraLite.NET supports ADO.NET development in the new `iAnywhere.Data.UltraLite` namespace. ADO.NET provides the benefits of an industry standard interface and of an easy migration path to Adaptive Server Anywhere for large applications. The UltraLite.NET component API (`iAnywhere.UltraLite` namespace) is deprecated in this release and will not be provided in the next major release.
- ◆ **Native UltraLite for Java component interface to be superseded by JDBC** The current Native UltraLite for Java interface is scheduled to be superseded by a JDBC interface.

Other behavior changes

The following is a list of behavior changes from previous versions of the software.

- ◆ **New warning for referential integrity deletes during download** UltraLite automatically deletes rows as needed to maintain referential

integrity during download. It now raises a warning for each row deleted in this way.

☞ For more information, see [“Referential integrity and synchronization”](#) [*Getting Started with MobiLink*, page 23].

- ◆ **Native UltraLite for Java behavior changes** `Cursor.getRowCount()` method has been changed to return an int. No application changes are required.
- ◆ **UltraLite.NET component behavior changes** `Cursor.getRowCount()` method has been changed to return an int. No application changes are required.
- ◆ **Handling invalid synchronization parameters** In previous releases, the UltraLite runtime ignored all invalid synchronization parameters. Misspelled parameters were therefore ignored and a default value used instead.

In this release, if the runtime encounters an invalid parameter, synchronization fails and the SQL code `SQLE_UNRECOGNIZED_OPTION` is set. If an error callback has been provided, it will be called once for each invalid parameter. Duplicates continue to be ignored.

- ◆ **New libraries for secure synchronization** The security options for synchronization have been moved into separate libraries. If you use either of the `ULSecureCerticomTLSStream` or `ULSecureRSATLSStream` security options for encrypted synchronization, you must now link separately against a corresponding static library, or ship a separate DLL.
- ◆ **UltraLite for MobileVB integration with Crossfire** If you have existing projects that use the UltraLite for MobileVB integration with Crossfire from an earlier version of the software, you must change the reference to `Interop.UltraLiteAFLib.dll` to `iAnywhere.UltraLiteForAppForge.dll`.

☞ For more information, see [“Tutorial: A Sample Application for AppForge Crossfire”](#) [*UltraLite for Crossfire User's Guide*, page 41].

QAnywhere behavior changes

Deprecated and discontinued features

- ◆ **QAnywhere Agent options** The following QAnywhere Agent (qaagent) options have been deprecated and replaced.

Deprecated qaagent option...	Replaced with qaagent option...
-agent_id <i>id</i>	-id <i>id</i>
-dbauser <i>user</i>	-c "UID=user"
-dbeng <i>name</i>	-c "ENG=name"
-dbfile <i>filename</i>	-c "DBF=filename"
-dbname <i>name</i>	-c "DBN=name"
-ek <i>key</i>	-c "DBKEY=key"
-password <i>password</i>	-c "PWD=password"
-sv	-c "Start={ dbeng9 dbsrv9 }"
-verbose	-v[levels]

In addition, the following qaagent options are no longer required and have been deprecated:

- ◆ **-e**
- ◆ **-rb**

For more information, see “qaagent syntax” [*QAnywhere User’s Guide*, page 123].

- ◆ **QAnywhere Agent no longer creates the client message store** You must now create the message store database yourself before running qaagent. There is a new option, **-si**, that initializes the database with system objects that are required by QAnywhere.

See “Setting up the client message store” [*QAnywhere User’s Guide*, page 43].

- ◆ **QAnywhere messages with an InReplyToID** The QAnywhere built-in header InReplyToID is no longer mapped to the JMS property `ias_ReplyToAddress`. This means that it is no longer copied to JMSCorrelationID.

☞ See “Mapping QAnywhere messages on to JMS messages” [*QAnywhere User’s Guide*, page 114].

Other behavior changes

- ◆ **QAnywhere client library message overhead is reduced** The `ias_MessageType` property is no longer set for regular messages. It is still set for network status and other system messages that are sent to the system queue.

CHAPTER 3

What's New in Version 9.0.1

About this chapter

This chapter provides an overview of the new features and behavior changes introduced in SQL Anywhere Studio version 9.0.1.

Contents

Topic:	page
New features in version 9.0.1	154
Behavior changes in version 9.0.1	172

New features in version 9.0.1

This section lists the new features introduced in components of SQL Anywhere Studio version 9.0.1.

Adaptive Server Anywhere new features

This section introduces the new features in Adaptive Server Anywhere version 9.0.1. It provides an exhaustive listing of major and minor new features, with cross references to locations where each feature is discussed in detail.

OLAP enhancements

- ◆ **OLAP query extensions** A set of online analytical processing (OLAP) features enable more detailed analysis of the data in your database. The enhancements include the ability to add subtotal rows into result sets in a flexible way using CUBE and using GROUPING SETS, as well as windowing functions that provide rolling averages and other advanced features.

Support for these OLAP features is built in to the Query Editor, which lets you build queries that use ROLLUP, CUBE, and GROUPING SETS operations.

☞ For more information, see the following:

- ◆ “OLAP Support” [*SQL Anywhere SQL User’s Guide*, page 401]
- ◆ “Using ROLLUP” [*SQL Anywhere SQL User’s Guide*, page 408]
- ◆ “Using CUBE” [*SQL Anywhere SQL User’s Guide*, page 410]
- ◆ “GROUP BY GROUPING SETS” [*SQL Anywhere SQL User’s Guide*, page 404]
- ◆ “Window aggregate functions” [*SQL Anywhere SQL User’s Guide*, page 422]
- ◆ **New statistical functions** Several statistical functions have been added.

For more information, see:

 - ◆ “COS function [Numeric]” [*SQL Anywhere Reference*, page 123]
 - ◆ “COVAR_POP function [Aggregate]” [*SQL Anywhere Reference*, page 125]
 - ◆ “COVAR_SAMP function [Aggregate]” [*SQL Anywhere Reference*, page 126]
 - ◆ “CUME_DIST function [Ranking]” [*SQL Anywhere Reference*, page 129]
 - ◆ “DENSE_RANK function [Ranking]” [*SQL Anywhere Reference*, page 142]
 - ◆ “PERCENT_RANK function [Ranking]” [*SQL Anywhere Reference*, page 192]

- ◆ “RANK function [Ranking]” [*SQL Anywhere Reference*, page 199]
- ◆ “REGR_AVGX function [Aggregate]” [*SQL Anywhere Reference*, page 200]
- ◆ “REGR_AVGY function [Aggregate]” [*SQL Anywhere Reference*, page 201]
- ◆ “REGR_COUNT function [Aggregate]” [*SQL Anywhere Reference*, page 203]
- ◆ “REGR_INTERCEPT function [Aggregate]” [*SQL Anywhere Reference*, page 204]
- ◆ “REGR_R2 function [Aggregate]” [*SQL Anywhere Reference*, page 205]
- ◆ “REGR_SLOPE function [Aggregate]” [*SQL Anywhere Reference*, page 206]
- ◆ “REGR_SXX function [Aggregate]” [*SQL Anywhere Reference*, page 207]
- ◆ “REGR_SXY function [Aggregate]” [*SQL Anywhere Reference*, page 209]
- ◆ “REGR_SYY function [Aggregate]” [*SQL Anywhere Reference*, page 210]
- ◆ “ROW_NUMBER function [Miscellaneous]” [*SQL Anywhere Reference*, page 216]

◆ **New string functions** The following string functions have been added:

- ◆ “BASE64_DECODE function [String]” [*SQL Anywhere Reference*, page 103]
- ◆ “BASE64_ENCODE function [String]” [*SQL Anywhere Reference*, page 104]
- ◆ “COMPRESS function [String]” [*SQL Anywhere Reference*, page 114]
- ◆ “DECOMPRESS function [String]” [*SQL Anywhere Reference*, page 139]
- ◆ “DECRYPT function [String]” [*SQL Anywhere Reference*, page 140]
- ◆ “ENCRYPT function [String]” [*SQL Anywhere Reference*, page 144]
- ◆ “HASH function [String]” [*SQL Anywhere Reference*, page 159]

Statement
enhancements

◆ **LOAD TABLE enhancements** The LOAD TABLE statement now has a clause that allows you to limit the statistics that are created, allowing faster table loading. It also has a SKIP option that allows you to ignore the first few lines of a file.




☞ For more information, see “LOAD TABLE statement” [*SQL Anywhere Reference*, page 558].

◆ **SELECT ... INTO base-table** This new SELECT syntax creates a base table and fills that table with data from a query.

☞ For more information, see “SELECT statement” [*SQL Anywhere Reference*, page 623].

- ◆ **Extended support for variables in SQL statements** Several statements have been made more flexible by permitting variables as well as constants in some locations. This is especially useful in stored procedures and batches, where variables can be declared and used. It provides functionality previously only available, in more cumbersome form, in EXECUTE IMMEDIATE.

The following statements have this extended support for variables:

- The TOP clause of the SELECT statement can now reference integer variables as well as constants. See “[SELECT statement](#)” [[SQL Anywhere Reference](#), page 623].
 - BACKUP statement *backup-directory* and *archive-root*. See “[BACKUP statement](#)” [[SQL Anywhere Reference](#), page 318].
 - RESTORE statement *filename*, *archive-root*, and new *dbspace-name*. See “[RESTORE DATABASE statement](#)” [[SQL Anywhere Reference](#), page 606].
 - LOAD TABLE statement *filename*. See “[LOAD TABLE statement](#)” [[SQL Anywhere Reference](#), page 558].
 - UNLOAD statement and UNLOAD TABLE statement *filename*. See “[UNLOAD statement](#)” [[SQL Anywhere Reference](#), page 671] and “[UNLOAD TABLE statement](#)” [[SQL Anywhere Reference](#), page 673].
- ◆ **SET statement enhancement** The SET statement now accepts the option `ansi_nulls` (equivalent to the `ansinull` option) for compatibility with Microsoft SQL Server.
 For information, see “[SET statement \[T-SQL\]](#)” [[SQL Anywhere Reference](#), page 633].
 - ◆ **ALTER TABLE statement enhancement** ALTER TABLE can now add a NOT NULL column with a default value to a non-empty table. This feature provides increased flexibility when modifying existing tables.
 For more information, see “[ALTER TABLE statement](#)” [[SQL Anywhere Reference](#), page 303].
 - ◆ **ALTER VIEW statement enhancements** The ALTER VIEW statement now supports a RECOMPILE clause that allows you to re-create view definitions when the columns in the underlying tables are modified.
 For more information, see “[ALTER VIEW statement](#)” [[SQL Anywhere Reference](#), page 314].
 - ◆ **MESSAGE statement enhancements** A FOR CONNECTION clause has been added to the MESSAGE statement.

Also, a `DEBUG ONLY` clause has been added to the `MESSAGE` statement. When the `debug_messages` option is set to `ON`, debugging messages appear for all stored procedures and triggers that contain a `MESSAGE` statement that includes the `DEBUG ONLY` clause.

☞ For more information, see “[MESSAGE statement](#)” [*SQL Anywhere Reference*, page 570] and “[debug_messages option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 390].

Security enhancements

- ◆ **Database page checksums** Database page checksums are used to detect whether a database page has been modified on disk. When a database is created with checksums enabled, a checksum is calculated for each page before it is written to disk. When a page is read from disk, its checksum is calculated again and compared to the stored checksum. If the values are different, the page has been modified or otherwise corrupted while on disk. Existing databases must be unloaded and reloaded into a database with checksums enabled to use this feature. You can check whether checksums are enabled for a database using the `Checksum` property.

☞ For more information about creating databases with checksums, see “[CREATE DATABASE statement](#)” [*SQL Anywhere Reference*, page 349], “[The Initialization utility](#)” [*SQL Anywhere Database Administration Guide*, page 573], and “[Database-level properties](#)” [*SQL Anywhere Database Administration Guide*, page 492].

Checksums can also be used to validate a database.

☞ For more information, see “[VALIDATE CHECKSUM statement](#)” [*SQL Anywhere Reference*, page 689], “[The Validation utility](#)” [*SQL Anywhere Database Administration Guide*, page 657], and “[sa_validate system procedure](#)” [*SQL Anywhere Reference*, page 923].

Performance enhancements

Many enhancements have been made to provide better performance for a wide range of tasks, including complex queries. Some of these enhancements are purely internal. Others are listed here:

- ◆ **Parallel index scans** On volumes with multiple disk spindles, such as hardware or software RAID arrays, the query optimizer can now scan tables using an index in parallel.

☞ For more information, see “[Parallel index scans](#)” [*SQL Anywhere SQL User's Guide*, page 529].
- ◆ **Clustered Hash Group By algorithm** For better performance, the Adaptive Server Anywhere query optimizer can use a new algorithm that is particularly useful for certain classes of `GROUP BY` queries where the `HAVING` clause returns a small proportion of rows.

☞ For more information, see “Clustered Hash Group By algorithm” [*SQL Anywhere SQL User’s Guide*, page 538], and “optimization_workload option [database]” [*SQL Anywhere Database Administration Guide*, page 427].

- ◆ **Database server cache warming** Three new database server command line options have been added to support cache warming. Cache warming is designed to help reduce the execution times of the initial queries executed against a database by pre-loading the database server’s cache with database pages that were referenced the last time the database was started. Using cache warming can improve performance when the same queries are executed against the database each time it is started.

☞ For more information, see “-cc server option” [*SQL Anywhere Database Administration Guide*, page 132], “-cr server option” [*SQL Anywhere Database Administration Guide*, page 135], “-cv server option” [*SQL Anywhere Database Administration Guide*, page 136], and “Using cache warming” [*SQL Anywhere SQL User’s Guide*, page 229].

- ◆ **Optimizer hints** WITH (XLOCK) is a new table hint feature in the FROM clause. XLOCK indicates that rows processed by the statement from the hinted table are to be locked exclusively. The affected rows remain locked until the end of the transaction. It works at all isolation levels.

The WITH INDEX hint forces the optimizer to use a specified index during query optimization. This is an advanced feature that may lead to poor performance if used incorrectly, and so should be used by experienced users only.

☞ For more information, see “FROM clause” [*SQL Anywhere Reference*, page 512] and “Use indexes for frequently-searched columns” [*SQL Anywhere SQL User’s Guide*, page 86].

- ◆ **Increased default stack size for internal execution threads on NetWare** The default stack size for internal execution threads on NetWare has been increased to 128 KB.

☞ For more information, see “-gss server option” [*SQL Anywhere Database Administration Guide*, page 153].

Programming interface enhancements

- ◆ **Perl interface** The Perl new DBD::ASAny driver for the Perl DBI module allows you to access and modify Adaptive Server Anywhere databases from Perl scripts.
- ◆ **InstallShield projects** SQL Anywhere studio now includes InstallShield Merge Module Projects and Object Projects. These projects allow InstallShield to generate Merge Modules and Objects with which you can redeploy the software currently installed on your computer.

Administration
enhancements

Previous versions of SQL Anywhere included the Merge Modules and Objects. These allowed you to redeploy the original software, but provided no convenient means of deploying after you had applied an EBF.

- ◆ **BACKUP enhancements** The BACKUP statement now includes an ON EXISTING ERROR clause for image backups. When this clause is specified, an error occurs if any of the files to be created during the backup already exist.

The archive backup form of the BACKUP statement has been extended to support options previously available only with image backups.

☞ For more information, see “[BACKUP statement](#)” [*SQL Anywhere Reference*, page 318].

The Backup utility can now create a backup on the server computer. Previously, the utility could only create backups on the client computer.

☞ “[The Backup utility](#)” [*SQL Anywhere Database Administration Guide*, page 546].

- ◆ **Unload utility (dbunload) enhancements** The Unload utility now automatically handles view dependencies when unloading databases. The -j option that was used in previous versions of the software to output view definitions multiple times to the *reload.sql* file has been deprecated. Now, the Unload utility automatically handles unloading view definitions that depend on other views.

The Unload utility also allows you to change the database page size when unloading into a new database.






☞ For more information, see “[Unload utility \(dbunload\)](#)” [*SQL Anywhere Database Administration Guide*, page 644].

- ◆ **Server Enumeration utility (dblocate) enhancements** The Server Enumeration utility (dblocate) now allows you to supply a host name or IP address to limit the search for database servers to a specific computer. As well, it supports a -n option that specifies that IP addresses are not to be resolved into computer names, which results in better performance.

☞ For more information, see “[The Server Enumeration utility](#)” [*SQL Anywhere Database Administration Guide*, page 609].

- ◆ **Adaptive Server Anywhere Console utility supports integrated logins** When you connect to the Adaptive Server Anywhere Console (dbconsole) utility on Windows NT/2000/XP, the Connect dialog allows you to use an integrated login to connect to the database.

☞ For more information, see “[Connect dialog: Identification tab \(SQL Anywhere\)](#)” [*SQL Anywhere 10 Help*, page 2].

- ◆ **The request log file size can be changed without restarting the database server** On starting the database server, you can specify the size of the request log file with the `-zs` server option. You can use the `sa_server_option` system procedure to change the size of the request log file without restarting the database server.
 For more information, see [“sa_server_option system procedure” \[SQL Anywhere Reference, page 907\]](#).
- ◆ **Additional information added for profiling system triggers** The `sa_procedure_profile` system procedure and `sa_procedure_profile_summary` system procedure now return extra information about system triggers when procedure profiling is turned on in the database.
 For more information, see [“sa_procedure_profile system procedure” \[SQL Anywhere Reference, page 895\]](#) and [“sa_procedure_profile_summary system procedure” \[SQL Anywhere Reference, page 898\]](#).
- ◆ **New system table** A new system table has been added that maintains information about the different versions of the software and platforms a database has been started with.
 For more information, see [“ISYSHISTORY system table” \[SQL Anywhere Reference, page 707\]](#).
- ◆ **New collations** There are two new collations available: one to support Lithuanian (1257LIT, ANSI Code Page 1257) and one to support Turkish (1254TRKALT). This Turkish collation considers I-dot and I-no-dot equal.
 For more information, see [“Supplied and recommended collations” \[SQL Anywhere Database Administration Guide, page 301\]](#) and [“Alternative Turkish collation 1254TRKALT” \[SQL Anywhere Database Administration Guide, page 290\]](#).
- ◆ **dedicated_task option** When specified, a request handling task is dedicated to handling requests from a single connection. This pre-established connection allows you to gather information about the state of the database server if it becomes otherwise unresponsive.
 For more information, see [“dedicated_task option \[database\]” \[SQL Anywhere Database Administration Guide, page 390\]](#).
- ◆ **Interactive SQL allows you to specify the encoding used to read and write files** The Interactive SQL READ, INPUT, and OUTPUT statements now support an optional encoding clause that allows you to specify the character encoding that is used to read or write the file. The `default_isql_encoding` option has been added to allow you to specify the

character encoding that is used for subsequent READ, INPUT, and OUTPUT statements.

☞ For more information, see “[default_isql_encoding option \[Interactive SQL\]](#)” [*SQL Anywhere Database Administration Guide*, page 391], “[READ statement \[Interactive SQL\]](#)” [*SQL Anywhere Reference*, page 593], “[INPUT statement \[Interactive SQL\]](#)” [*SQL Anywhere Reference*, page 542], and “[OUTPUT statement \[Interactive SQL\]](#)” [*SQL Anywhere Reference*, page 577].

You can also specify the character encoding used to read or write the file when using the Interactive SQL import and export wizards.

☞ For more information, see “[Importing and Exporting Data](#)” [*SQL Anywhere SQL User's Guide*, page 659] and “[Export wizard](#)” [*SQL Anywhere 10 Help*, page 233].

- ◆ **Interactive SQL supports integrated logins** When you connect to Interactive SQL on Windows NT/2000/XP, the Connect dialog allows you to use an integrated login to connect to the database.

☞ For more information, see “[Connect dialog: Identification tab \(SQL Anywhere\)](#)” [*SQL Anywhere 10 Help*, page 2].

- ◆ **Interactive SQL allows you to configure the font used for displaying result sets** You can choose the font, font style, and point size for data that appears in the Results pane in Interactive SQL.

☞ For information about configuring the Results pane in Interactive SQL, see “[Options dialog: Results tab](#)” [*SQL Anywhere 10 Help*, page 222].

- ◆ **Interactive SQL allows you to specify the initial folder used for file browsing** When browsing for files in Interactive SQL, you can specify whether Interactive SQL uses the current directory (as defined by the operating system) for the initial directory, or the last folder where a file was opened.

☞ For more information, see “[Options dialog: General tab](#)” [*SQL Anywhere 10 Help*, page 220].

Sybase Central enhancements

- ◆ **Sybase Central allows you to configure the font used for displaying result sets** You can choose the font, font style, and point size for data that appears on the Data tab in Sybase Central when a table is selected.

☞ For information about configuring the Data tab in Sybase Central, see “[Plug-in Preferences dialog: Table Data tab](#)” [*SQL Anywhere 10 Help*, page 134].

- ◆ **Remote Server Creation wizard now supports creating external login for current user** The Remote Server Creation wizard now allows you

to create an external login for the current user so that you do not have to create an external login before you create the remote server.

☞ For more information, see [“Creating remote servers using Sybase Central” \[SQL Anywhere SQL User’s Guide, page 709\]](#).

- ◆ **Sybase Central supports integrated logins** When you connect to Sybase Central on Windows NT/2000/XP, the Connect dialog allows you to use an integrated login to connect to the database.

☞ For more information, see [“Connect dialog: Identification tab \(SQL Anywhere\)” \[SQL Anywhere 10 Help, page 2\]](#).

- ◆ **Columns can be sorted using the View menu in Sybase Central**

The Sybase Central View menu has a Sort item that allows you to sort columns in the right pane as an alternative to clicking the column headings in the right pane.

- ◆ **Foreign key settings can be modified from the foreign key property sheet** You can change foreign key settings in Sybase Central from the Foreign Key property sheet.

☞ For more information, see [“Change Settings dialog” \[SQL Anywhere 10 Help, page 119\]](#).

- ◆ **Proxy Table wizard now displays primary key column information**

Previously, when creating a proxy table using the Proxy Table wizard, there was no way to determine which columns belonged to the remote table’s primary key. Now, the columns in the primary key are identified in the wizard.

- ◆ **Utility wizards can be cancelled** The Upgrade Database wizard, Backup Database wizard, Restore Database wizard, Validate Database wizard, Compress Database wizard, Uncompress Database wizard, and the Create Backup Images wizard can be cancelled. They also include a messages dialog that displays status information about whether the operation has succeeded or failed.

- ◆ **Sybase Central supports account names of the form domain\user when creating and editing services** The Service Creation wizard and Service property sheet now allow you to enter account names of the form **domain\user** when creating and editing services. You can enter the account name in the Other Account field on the Account tab of the Service property sheet or in the Service Creation wizard.

Miscellaneous
enhancements

- ◆ **Database server uses asynchronous I/O on Linux platforms** When running the database server on Linux, the database server uses

asynchronous I/O by default when possible. The `-ua` database server option allows you to turn off the use of asynchronous I/O.

☞ For more information, see “[-ua server option](#)” [*SQL Anywhere Database Administration Guide*, page 175].

- ◆ **openxml supports equality predicates** The `openxml` function allows you to use equality predicates in the XPath expression. This feature allows you to locate nodes within the XML document using attribute values.

☞ For more information, see “[openxml system procedure](#)” [*SQL Anywhere Reference*, page 831].

- ◆ **TransactionStartTime connection property** This property returns the time the database was first modified after a COMMIT or ROLLBACK.

☞ For more information, see “[Connection-level properties](#)” [*SQL Anywhere Database Administration Guide*, page 458].

- ◆ **UserAppInfo property** This property returns the portion of a connection string specified with the AppInfo connection parameter.

☞ For more information, see “[Connection-level properties](#)” [*SQL Anywhere Database Administration Guide*, page 458].

- ◆ **ConsoleLogFile server property** This property returns the name of the file where messages from the Server Messages window are logged when the `-o` server option is specified.

☞ For more information, see “[Server-level properties](#)” [*SQL Anywhere Database Administration Guide*, page 481].



- ◆ **DriveType database property for UNIX platforms** The DriveType database property has been extended to UNIX platforms.

☞ For more information, see “[Database-level properties](#)” [*SQL Anywhere Database Administration Guide*, page 492].

- ◆ **Connection IDs start at 1 and are incremented for each new connection to the database server** When the database server is started, each connection to the server is assigned a connection ID, starting with 1, and the connection number is incremented with each new connection to the server. The connection IDs are logged in the `-z` server output and the LogFile connection parameter output. They are also used by the CONNECTION_PROPERTY, NEXT_CONNECTION, NEXT_DATABASE, and DROP CONNECTION functions, and by request logging.




☞ For more information, see “[CONNECTION_PROPERTY function \[System\]](#)” [*SQL Anywhere Reference*, page 117], “[NEXT_CONNECTION](#)

function [System]” [*SQL Anywhere Reference*, page 185],
“NEXT_DATABASE function [System]” [*SQL Anywhere Reference*,
page 186], and “Request logging” [*SQL Anywhere SQL User’s Guide*,
page 203].

- ◆ **Improved cache management on NetWare and UNIX** When the cache size specified with -c is greater than the amount of available memory on UNIX or NetWare, the database server now calculates the maximum cache size based on available memory.
 For more information about how the database server calculates the maximum cache size in these circumstances, see “-c server option” [*SQL Anywhere Database Administration Guide*, page 129].
- ◆ **odbc_distinguish_char_and_varchar option** The odbc_distinguish_char_and_varchar option controls how the Adaptive Server Anywhere ODBC driver describes CHAR columns.
 For more information, see “odbc_distinguish_char_and_varchar option [database]” [*SQL Anywhere Database Administration Guide*, page 420].

MobiLink new features

Following is a list of changes and additions to the software introduced in version 9.0.1.

- ◆ **QAnywhere messaging** MobiLink QAnywhere provides application-to-application messaging capabilities. It allows you to write applications that exchange messages with remote applications located on a variety of devices running on Windows or Windows CE operating systems.
 For more information, see *QAnywhere User’s Guide*.
- ◆ **External authentication** MobiLink user authentication has been enhanced so that you can easily authenticate users using LDAP servers and other external sources such as POP3 email servers.
 For more information, see “Authenticating to external servers” [*MobiLink Clients*, page 22].
- ◆ **New MobiLink system tables** There are several new MobiLink system tables. The existence of new MobiLink system tables means that you need to upgrade your Adaptive Server Anywhere databases and run upgrade scripts for other consolidated databases.
 For more information, see “MobiLink System Tables” [*MobiLink Administration Guide*, page 595].

Adaptive Server
Anywhere client
enhancements

- ◆ **Configurable script versions** You can use the new `ml_property` MobiLink system table to store properties for script versions.
 ☞ For more information, see “[ml_property](#)” [*MobiLink Administration Guide*, page 607] and “[ml_add_property](#)” [*MobiLink Administration Guide*, page 576].
- ◆ **iAnywhere ODBC drivers** There is now an iAnywhere ODBC driver for DB2 available on Windows. This is a wire protocol driver, so DB2 client software is not required.
- ◆ **IBM DB2 setup scripts have version numbers** Multiple versions of the MobiLink synchronization server can now use the same DB2 server instance. This is possible because the two Java class files that MobiLink uses for DB2 stored procedures now include the SQL Anywhere Studio version number. For the 9.0.1 release they are called *SyncDB2_901.class* and *SyncDB2Long_901.class*.
 ☞ For more information, see “[IBM DB2 UDB consolidated database](#)” [*MobiLink Administration Guide*, page 14].
- ◆ **New -us server option** A new server option improves performance by preventing MobiLink from invoking unnecessary table scripts.
- ◆ **ActiveSync provider now generates an activity log file** The ActiveSync provider can now generate a log of its activities.
 ☞ For more information, see “[ActiveSync provider installation utility \[mlasinst\]”](#) [*MobiLink Clients*, page 28].
- ◆ **Improved application integration for Adaptive Server Anywhere clients** A new integration component for `dbmlsync` provides an easier and more customizable way to create applications with Adaptive Server Anywhere remote databases on Windows platforms.
 ☞ For more information, see “[Dbmlsync Integration Component](#)” [*MobiLink Clients*, page 279].
- ◆ **Resuming failed downloads** You can now avoid lengthy retransmission of data when downloads fail for both Adaptive Server Anywhere and UltraLite remote databases. After a partial transmission of a download, failed downloads may be resumed.
 ☞ For more information, see “[Resuming failed downloads](#)” [*MobiLink Administration Guide*, page 141].
- ◆ **Transaction-level uploads for Adaptive Server Anywhere clients** You can now choose to preserve transactions on the remote database in your upload, and you can do this per synchronization.
 ☞ For more information, see “[-tu option](#)” [*MobiLink Clients*, page 186].

Server-initiated
synchronization

- ◆ **New way to specify extended options** You can use the new hook `sp_hook_dbmlsync_set_extended_options` to programmatically customize the behavior of an upcoming synchronization.
For more information, see “[sp_hook_dbmlsync_set_extended_options](#)” [*MobiLink Clients*, page 262].
- ◆ **Automatic device tracking** Automated device tracking simplifies the Notification process.
☞ For more information, see “[Device tracking](#)” [*MobiLink Server-Initiated Synchronization User’s Guide*, page 23].
- ◆ **Sybase Central configuration** You can now configure notification using the Sybase Central MobiLink plug-in.
☞ For more information, see “[Setting properties](#)” [*MobiLink Server-Initiated Synchronization User’s Guide*, page 15].
- ◆ **Optional delivery confirmation** You can now configure your notification so that a confirmation is automatically sent to the consolidated database when a message is received.
☞ For more information, see “[Listener syntax](#)” [*MobiLink Server-Initiated Synchronization User’s Guide*, page 42].
- ◆ **New Listener options** The Listener has several new command line options, including the ability to listen on more than one channel at a time.
☞ For more information, see “[Listener Utility](#)” [*MobiLink Server-Initiated Synchronization User’s Guide*, page 41].
- ◆ **Palm configuration** There is now a Palm Listener configuration utility (`dblsncfg`) that simplifies configuration of Palm devices.
☞ For more information, see “[Listeners for Palm Devices](#)” [*MobiLink Server-Initiated Synchronization User’s Guide*, page 55].

MobiLink Monitor

- ◆ **Exporting Monitor data** Monitor data can now be exported to a relational database or Excel file.
- ◆ **Enhanced information in Monitor** You can now customize which columns are shown in the table. In addition, there are new columns containing information that was previously available only from synchronization property sheets, and a new column to uniquely identify synchronizations for a Monitor session.
- ◆ **Improved sorting** Sorting for the Monitor table has been improved. The sort order is now maintained when data is added or updated.

- ◆ **Enhanced user interface** There are new menus and tool bar buttons to zoom in, zoom out, or zoom to selection in the chart. In addition, pausing now controls whether the chart automatically scrolls.

☞ For more information, see “[MobiLink Monitor](#)” [*MobiLink Administration Guide*, page 161].

New web server support

- ◆ **Redirector supports Apache web servers** There is a new native Redirector for Apache web servers. Tomcat is no longer a requirement if you want to use an Apache web server.

☞ For more information, see “[Synchronizing Through a Web Server with the Redirector](#)” [*MobiLink Administration Guide*, page 181].

UltraLite new features

UltraLite 9.0.1 introduces several new features:

- ◆ **UltraLite.NET controls** A set of controls are added to your Visual Studio.NET 2003 toolbox to make it easier to specify connection parameters and to monitor synchronization from UltraLite.NET applications.

☞ For more information, see “[Tutorial: Build an UltraLite.NET Application](#)” [*UltraLite.NET User's Guide*, page 25].

- ◆ **UltraLite for M-Business Anywhere.** A new component is available for iAnywhere M-Business Anywhere, previously known as AvantGo M-Business Server.





☞ For more information, see [UltraLite for M-Business Anywhere User's Guide](#).

- ◆ **CREATE and DROP statements in dynamic SQL** CREATE/DROP TABLE and CREATE/DROP INDEX statements are now available in dynamic SQL. For users of UltraLite components, these statements provide a way of changing the schema of an UltraLite database.

☞ For more information, see “[CREATE INDEX statement](#)” [*UltraLite Database User's Guide*, page 290], and “[CREATE TABLE statement](#)” [*UltraLite Database User's Guide*, page 293].

- ◆ **Transaction control from dynamic SQL** COMMIT and ROLLBACK statements are now available in dynamic SQL. For users of UltraLite components, these statements provide a way of using SQL statements to control transactions. They provide an alternative to the commit and rollback methods on the connection object.

☞ For more information, see “[COMMIT statement](#)” [*UltraLite Database User's Guide*, page 289], and “[ROLLBACK statement](#)” [*UltraLite Database User's Guide*, page 303].

- ◆ **Dynamic SQL SELECT enhancements** Subqueries can be used in search conditions in the WHERE clause or the HAVING clause. They can also be used as derived tables in the FROM clause.
 For more information, see “[SELECT statement](#)” [*UltraLite Database User's Guide*, page 304] and “[Search conditions](#)” [*UltraLite Database User's Guide*, page 204].
The HAVING clause is now supported. See “[SELECT statement](#)” [*UltraLite Database User's Guide*, page 304].
- ◆ **ODBC interface to UltraLite** UltraLite now supports a subset of the ODBC programming interface.
- ◆ **Mixing C++ interfaces** The UltraLite C/C++ based interfaces (embedded SQL, the static C++ API, and the C++ Component) may be used in the same application.
Of particular interest is adding C++ Component dynamic SQL to an existing embedded SQL or static C++ API application, or using embedded SQL to execute general SQL in a primarily C++ Component-based application.
 The new functions include “[db_start_database function](#)” [*UltraLite C/C++ User's Guide*, page 285] (embedded SQL) and StartDatabase method.
- ◆ **CodeWarrior stationery for UltraLite C++ component** The stationery **Palm OS UltraLite C++ Component App** is provided as part of the UltraLite plug-in for CodeWarrior. It assists in building C++ Component applications using CodeWarrior for Palm OS.
The files for the UltraLite plug-in for CodeWarrior are placed on your disk during UltraLite installation, but the plug-in is not available for use without an additional installation step.
 For more information, see “[Developing UltraLite applications with Metrowerks CodeWarrior](#)” [*UltraLite C/C++ User's Guide*, page 81].
- ◆ **Improved UltraLite C/C++ error handling** An error callback is now supported for all UltraLite C/C++ interfaces. The callback allows the application to be notified of all errors, and so provides developers with invaluable information during development.
 For more information, see “[ULRegisterErrorCallback function](#)” [*UltraLite C/C++ User's Guide*, page 178] and “[Callback function for ULRegisterErrorCallback](#)” [*UltraLite C/C++ User's Guide*, page 156].
- ◆ **UltraLite components can use the engine** The UltraLite database engine, which can accept connections from more than one application, is

now available to UltraLite components as an alternative deployment option.

This option is not available to UltraLite for MobileVB or UltraLite ActiveX.

- ◆ **Database conversion tool** The `ulconv` utility is a command line tool for carrying out numerous operations on UltraLite databases, including unloading databases to XML files, loading new databases from XML files, and converting database formats.
- ◆ **Additional synchronization progress event** An additional event is available to the synchronization observer, when an error has occurred and the downloaded changes are being rolled back.

For more information, see the `ULSyncState` structure or object for the API you are using.

- ◆ **Schema upgrade monitoring** Schema upgrades can be a long operation. New schema upgrade events provides a mechanism for applications to monitor the progress of a schema upgrade.
- ◆ **Restartable downloads** UltraLite can now restart downloads that fail due to communications errors or user cancellation through the synchronization observer.

☞ For more information, see [“Resuming failed downloads” \[MobiLink Administration Guide, page 141\]](#).

- ◆ **New Windows CE platform support** UltraLite now supports the Smartphone 2002 platform. ActiveSync synchronization is not supported on this platform.

UltraLite also supports Windows CE 4.1 on ARM chips in V4T (“thumb”) mode.

- ◆ **Multi-database support** UltraLite components can address multiple databases from a single application by issuing multiple connection requests specifying different database filenames or creator IDs.

There are some extensions to the connection parameters as a result of this feature.

☞ For more information, see [“DBN connection parameter” \[UltraLite Database User's Guide, page 127\]](#).

- ◆ **ULPalmLaunch and ULPalmExit no longer required** UltraLite now supports additional connection-related primitives that provide easier support for maintaining state when an application is closed. As a result of

these new features, Palm OS applications no longer require special Palm-specific primitives, including ULPalmLaunch and ULPalmExit.

☞ For more information, see “[Saving state in UltraLite Palm applications](#)” [*UltraLite C/C++ User’s Guide*, page 86], “[Maintaining state in UltraLite Palm applications](#)” [*UltraLite for Crossfire User’s Guide*, page 37] and “[Connecting to a database](#)” [*UltraLite C/C++ User’s Guide*, page 40].

- ◆ **UltraLite database properties** Properties of the UltraLite database are now available to UltraLite component applications. The case sensitivity, collation, and database ID used for global autoincrement values are all available as properties or methods of the Connection object, depending on the API.


Documentation enhancements

This section introduces enhancements made to the appearance, organization, or navigation of the Adaptive Server Anywhere documentation for version 9.0.1. It provides an exhaustive listing of major changes.

New documentation

- ◆ **MobiLink Server-Initiated Synchronization** The MobiLink server-initiated synchronization feature has been moved into its own book.
☞ For more information, see [MobiLink Server-Initiated Synchronization User’s Guide](#).
- ◆ **QAnywhere Messaging** There is now a book describing the new MobiLink messaging application, QAnywhere.
☞ For more information, see [QAnywhere User’s Guide](#).
- ◆ **New MobiLink tutorials** There are new tutorials describing how to use Java and .NET scripting logic with Adaptive Server Anywhere remote databases.
☞ For more information, see “[Tutorial: Using Java Synchronization Logic](#)” [*Getting Started with MobiLink*, page 81] and “[Tutorial: Using .NET Synchronization Logic](#)” [*Getting Started with MobiLink*, page 95].
- ◆ **New chapter describing the dbmlsync integration component** For more information, see “[Dbmlsync Integration Component](#)” [*MobiLink Clients*, page 279].
- ◆ **Combined UltraLite C/C++ books** The documentation for UltraLite C/C++ interfaces (embedded SQL, Static C++ API, and the C++ Component) has been combined into a single book, together with the new ODBC interface.
☞ See [UltraLite C/C++ User’s Guide](#).

Documentation
enhancements

- ◆ **UltraLite for M-Business Anywhere** The new UltraLite for M-Business Anywhere component has its own book.
 See *UltraLite for M-Business Anywhere User's Guide*.
- ◆ **Adaptive Server Anywhere Getting Started and Introducing SQL Anywhere Studio books merged** The Adaptive Server Anywhere Getting Started and Introducing SQL Anywhere Studio books have been merged into one book. The new book is called *Introducing SQL Anywhere Studio*.
- ◆ **Enhanced PDF books** Some people find PDF to be a useful alternative to HTML-based online books, particularly for conceptual material. The PDF version of the online books is now installed by default, and can be accessed from the Windows start menu or from the online books by clicking the PDF item at the top of each topic.

The PDF files feature clickable links not only within a book, but to other books and to material on web sites. The behavior of these features depends on whether you read the files from within a browser plugin or directly from Acrobat Reader. For the best experience, use Acrobat Reader directly.
- ◆ **Help on help** A new section describes the differences between the online books in Windows HTML Help format, and the online books in PDF format. It also describes how to use the various help features to navigate the documentation and access the information you are looking for.

Behavior changes in version 9.0.1

This section lists the behavior changes introduced in components of SQL Anywhere Studio version 9.0.1.

Adaptive Server Anywhere behavior changes


Deprecated and discontinued features

This list includes features that are no longer supported or are deprecated, and that may impact existing applications.

- ◆ **MDSR encryption discontinued** Previously, both MDSR and AES strong encryption were supported in Adaptive Server Anywhere. Currently, the only type of strong encryption now supported is AES encryption. This change means that the **-ea** option is no longer required when using encryption with the Initialization utility, the Extract utility, or the Unload utility. It also removes the *algorithm* parameter from both the Create Database statement and the Create Encrypted File statement.
- ◆ **Write files deprecated** The use of write files is deprecated with this release.
- ◆ **Compressed database files deprecated** The use of compressed database files is deprecated with this release.
- ◆ **Unload utility (dbunload) -j option deprecated** As a result of enhancements to the Unload utility, the -j Unload utility option is no longer supported.
- ◆ **Language Selection utility (dblang) -d option deprecated** The -d option for the Language Selection utility, which was used to change the Adaptive Server Anywhere registry setting, is no longer supported.

Other behavior changes

The following is a list of behavior changes from previous versions of the software.

- ◆ **CURRENT_TIMESTAMP and CURRENT_USER special values added**
The CURRENT_TIMESTAMP (equivalent to CURRENT_TIMESTAMP) and CURRENT_USER (equivalent to CURRENT_USER) special values have been added for compatibility with Microsoft SQL Server.
 For more information, see [“CURRENT_TIMESTAMP special value” \[SQL Anywhere Reference, page 32\]](#) and [“CURRENT_USER special value” \[SQL Anywhere Reference, page 32\]](#).
- ◆ **db_start_engine returns SQLCODE 0 if database server is already running** db_start_engine now returns non-zero and sets the SQLCODE to 0 if the database server is already running. Previously, db_start_engine

returned non-zero, but set the SQLCODE to SQLE_ENGINE_ALREADY_RUNNING.

☞ For more information, see “[db_start_engine function](#)” [*SQL Anywhere Programming Guide*, page 448].

- ◆ **db_start_database returns non-zero and SQLCODE 0 if database is already running** db_start_database now returns non-zero and set the SQLCODE to 0 if the database is already running. Previously, db_start_database returned 0 (indicating failure) and set the SQLCODE to SQLE_ALIAS_CLASH.

☞ For more information, see “[db_start_database function](#)” [*SQL Anywhere Programming Guide*, page 447] and “[db_start_database function](#)” [*UltraLite C/C++ User's Guide*, page 285].

- ◆ **Default algorithm changed for the validation utility** The Validation utility (dbvalid) now uses the express check -fx option) algorithm by default. The express check algorithm performs significantly faster for large tables that have several indexes where the table does not completely fit in the server's cache. You can specify the -fn option if you want to use the validation algorithm that was used in previous versions of Adaptive Server Anywhere.

☞ For more information, see “[Validation utility \(dbvalid\)](#)” [*SQL Anywhere Database Administration Guide*, page 658].

- ◆ **Entering multibyte characters in the Connect dialog** In order to conform to Windows security practices, you can no longer use an Input Method Editor (IME) to type Japanese and other Asian language multibyte characters in the Password field of the Connect dialog used in Sybase Central, Interactive SQL, and the Adaptive Server Anywhere Console utility.

If you have an existing database that includes these characters in passwords, you can type your password in the Additional Connection Parameters field on the Advanced tab of both the Connect dialog and the ODBC Configuration dialog. However, you should note that when the password is typed on the Advanced tab, it is not obscured and is visible in plain text. When upgrading your database, it is recommended that you change your passwords so they do not include multibyte characters.

- ◆ **Owner name cannot be specified in DECLARE LOCAL TEMPORARY TABLE statements** In previous versions of the software, if an owner name was specified in DECLARE LOCAL TEMPORARY TABLE and the owner was not the same as the current user, it was possible to create more than one temporary table with the same name. A syntax error now occurs if an owner name is specified.

- ◆ **min_table_size_for_histogram option default setting changed** The min_table_size_for_histogram option specifies the minimum table size for which histograms are created. The default value has been changed to 100 rows. In previous versions of the software, the default value was 1000 rows. This setting can be changed in databases created with earlier versions of the software using the SET OPTION statement.
- ◆ **NULL constants data type conversion change** In previous versions, when converting a NULL constant to a CHAR, VARCHAR, LONG VARCHAR, BINARY, VARBINARY, or LONG BINARY type, the size of the column would be initialized to 32767 if no length was provided. Now, the size is initialized to 0.



For example, the following queries previously returned a column described as length 32767:

```
SELECT CAST( NULL AS CHAR )
-- This now returns a CHAR(0) column

SELECT 'abc'
UNION ALL
SELECT NULL
-- This now returns a CHAR(3) column


SELECT ''
UNION ALL
SELECT NULL
-- This now returns a CHAR(0) column

SELECT IF 1=1 THEN 'abc' ELSE NULL ENDIF
-- This now returns a CHAR(3) column
```

- ◆ **UPDATE statements and errors when ORDER BY clauses use ordinal values** UPDATE statements containing an ORDER BY clause that uses ordinal values now return a syntax error.
- ◆ **Restrictions on identifiers** You can no longer use double backslashes or double quotes in identifiers. Backslashes are permitted in identifiers only if used as an escape character.
 For more information, see “[Identifiers](#)” [*SQL Anywhere Reference*, page 8].
- ◆ **EXECUTE IMMEDIATE statement default setting WITH RESULT SET clause** The EXECUTE IMMEDIATE statement can return a result set when you specify the WITH RESULT SET ON clause. The default setting is WITH RESULT SET OFF.
 For more information, see “[EXECUTE IMMEDIATE statement \[SP\]](#)” [*SQL Anywhere Reference*, page 496].

MobiLink behavior changes


The following is a list of behavior changes from previous versions of the software.

- ◆ **Change in the order hooks are called** The order in which event hooks are called has been changed. This means that incremental uploads are now more efficient, as the upload events are the only part of the synchronization sequence that is repeated for each incremental update.
 For more information, see [“Synchronization event hook sequence”](#) [*MobiLink Clients*, page 201].
- ◆ **Fix to row-wise partitioning for publications for Adaptive Server Anywhere clients** Publications containing a WHERE clause now only replicate rows that meet the WHERE condition. From versions 8.0.0 through 9.0.0, a bug existed that caused rows to be replicated when the WHERE clause evaluated to an unknown value. For example, if a publication WHERE clause had “WHERE val = 1”, rows where val was NULL would also be replicated. This bug affected both SQL Remote and Adaptive Server Anywhere MobiLink clients.

UltraLite behavior changes

The following is a list of behavior changes from previous versions of the software.

- ◆ **Palm OS state management** For Palm OS applications using embedded SQL or the static C++ API, the `ULPalmExit` (`ULData::PalmExit`) and `ULPalmLaunch` (`ULData::PalmLaunch`) functions are no longer needed to manage state and synchronization information, and are now deprecated. The `ULData` and `ULConnection` `Reopen` methods are also deprecated.

Applications on Palm OS now use the same sequence of initialization, connection, and closing functions as other applications. The `ULSetSynchInfo` method controls `HotSync` synchronization.
 For more information, see [“Adding HotSync synchronization to Palm applications”](#) [*UltraLite C/C++ User's Guide*, page 92].
- ◆ **Palm OS 3.0 no longer supported** The earliest version of supported in this release is Palm OS 3.5.
- ◆ **ULEnableGenericSchema function deprecated** UltraLite C/C++ applications that require schema upgrades no longer need to call

ULEnableGenericSchema. Instead, use the function
ULRegisterSchemaUpgradeObserver.

- ◆ **UltraLite components Table API** The Delete method of the Table object no longer automatically refreshes the row after deleting. To maintain previous behavior, refetch the row using Relative(0) after the Delete operation.
- ◆ **Native UltraLite for Java casting of column IDs and parameter IDs no longer required** All methods that accepted column IDs and parameter IDs, and some methods that accepted other short-typed parameters have been changed to accept integers. This eliminates the need for casting numeric constants in your code. For example, instead of
`table.getString((short)1);` you can now use
`table.getString(1);`.

As a result of this change, Native UltraLite for Java applications must be recompiled to work with 9.0.1 software. No code changes are required.

CHAPTER 4

What's New in Version 9.0.0

About this chapter

This chapter provides an overview of the new features and behavior changes introduced in SQL Anywhere Studio version 9.

Contents

Topic:	page
New features in version 9.0.0	178
Behavior changes in version 9.0	201

New features in version 9.0.0

This section lists the new features introduced in components of SQL Anywhere Studio version 9.0.

Adaptive Server Anywhere new features

This section introduces the new features in Adaptive Server Anywhere version 9.0. It provides an exhaustive listing of major and minor new features, with cross references to locations where each feature is discussed in detail.

Highlighted new features

- ◆ **XML support** Adaptive Server Anywhere 9.0.0 includes a broad range of support for XML, including storing XML documents, exporting relational data as XML, importing XML, and returning XML from queries on relational data.
 - **FOR XML clause** The SELECT statement supports a FOR XML clause with three modes, RAW, AUTO, and EXPLICIT, that allow you to obtain query results as an XML document. Each mode allows you a different level of control over the format of the XML that is generated.
☞ For more information, see [“Obtaining query results as XML” \[SQL Anywhere SQL User’s Guide, page 628\]](#) and [“SELECT statement” \[SQL Anywhere Reference, page 623\]](#).
 - **for_xml_null_treatment option** You can use the for_xml_null_treatment option to control how NULL values are returned by a query that includes the FOR XML clause.
☞ For more information, see [“for_xml_null_treatment option \[database\]” \[SQL Anywhere Database Administration Guide, page 398\]](#).
 - **openxml procedure** For more information, see [“openxml system procedure” \[SQL Anywhere Reference, page 831\]](#).
 - **SQL/XML support** SQL/XML is a draft standard that describes the ways SQL can be used in conjunction with XML. As part of its SQL/XML support, Adaptive Server Anywhere includes an XML data type that can be used to store XML documents in the database.
☞ For more information, see [“XML data type” \[SQL Anywhere Reference, page 55\]](#).
- Adaptive Server Anywhere also supports the following SQL/XML functions that provide an alternative method to the FOR XML clause for generating XML documents from your relational data:
 - ◆ **XMLAGG function** This aggregate function generates a forest of XML elements from a collection of XML elements.
☞ For more information, see [“XMLAGG function \[Aggregate\]” \[SQL Anywhere Reference, page 249\]](#).

- ◆ **XMLCONCAT function** This function generates a forest of XML elements by concatenating together the XML values that are passed in to it.
☞ For more information, see “[XMLCONCAT function \[String\]](#)” [*SQL Anywhere Reference*, page 250].
- ◆ **XMLELEMENT function** This function generates an XML element for which you can optionally specify element content, attributes, and attribute content.
☞ For more information, see “[XMLELEMENT function \[String\]](#)” [*SQL Anywhere Reference*, page 250].
- ◆ **XMLFOREST function** This function generates a forest of XML elements.
☞ For more information, see “[XMLFOREST function \[String\]](#)” [*SQL Anywhere Reference*, page 252].
- ◆ **XMLGEN function** This function generates an XML value based on an XQuery Constructor.
☞ For more information, see “[XMLGEN function \[String\]](#)” [*SQL Anywhere Reference*, page 253].

- ◆ **HTTP server in the database** Adaptive Server Anywhere database servers can now act as web servers, allowing you to write and run web-based applications using only an Adaptive Server Anywhere database and a web browser of your choice.

This feature allows the database server to handle standard HTTP and HTTPS requests, as well as standard SOAP requests. Service types available are HTTP, HTTPS, XML, RAW, SOAP, and DISH. DISH is a SOAP service handler.

To gain the benefits of this enhancement on databases created before this release, you must upgrade the database using the Upgrade utility.

☞ For more information, see “[SQL Anywhere Web Services](#)” [*SQL Anywhere Programming Guide*, page 511].

- ◆ **Index Consultant** The Index Consultant is a tool to assist you in proper selection of indexes. It analyzes either a single query or a set of operations, and recommends indexes to add to your database and to remove from the database.

To gain the benefits of this enhancement on databases created before this release, you must upgrade the database using the Upgrade utility.

☞ For more information, see “[Index Consultant overview](#)” [*SQL Anywhere SQL User's Guide*, page 90].

- ◆ **64-bit version available** A full 64-bit version of the software is available for Windows Server 2003 on Itanium II chips. A deployment

SQL enhancements

release is available on 64-bit Linux and HP-UX operating systems.

- ◆ **The WITH clause can now be used before a select to specify common table expressions** Common table expressions are temporary view definitions that exist only within the scope of a SELECT statement. They can be recursive, or non-recursive. They sometimes let you write queries in a more elegant manner. They also permit you to perform multiple levels of aggregation within a single query. They can be used only within a top-level SELECT statement, within the top-level SELECT statement within a view definition, or within the top-level statement within an INSERT statement.

For more information, see [“Common Table Expressions” \[SQL Anywhere SQL User’s Guide, page 379\]](#).

- ◆ **Recursive union can now be performed using a common table expression of a particular form** Recursive common table expressions allow you to write recursive queries. These are particularly useful when querying tables that represent hierarchical data structures or directed graphs. Each recursive common table expression contains an initial subquery, which is executed first, and a recursive subquery. The A reference to the view, which must appear within the FROM clause of the recursive subquery, references the rows added to the view during the previous iteration. You must be particularly careful to provide conditions that stop the recursion if the data structure you are querying may contain cycles.

For more information, see [“Recursive common table expressions” \[SQL Anywhere SQL User’s Guide, page 388\]](#).

- ◆ **INTERSECT and EXCEPT operations are now supported** These operations compute the intersection and difference between two or more result sets. They complement the UNION operation.

For more information, see the following:

- [“Using EXCEPT and INTERSECT” \[SQL Anywhere SQL User’s Guide, page 324\]](#)
- [“Set operators and NULL” \[SQL Anywhere SQL User’s Guide, page 326\]](#)
- [“EXCEPT operation” \[SQL Anywhere Reference, page 491\]](#)
- [“INTERSECT operation” \[SQL Anywhere Reference, page 554\]](#)

- ◆ **SELECT statements can operate on stored procedure result sets** In SELECT statements, a stored procedure call can now appear anywhere a base table or view is allowed.

If you want statistics on stored procedure calls to be stored, you must upgrade the database using the Upgrade utility. Without statistics, you may get bad plans if you try to join the result of a stored procedure call.

For more information, see [“FROM clause” \[SQL Anywhere Reference, page 512\]](#).

- ◆ **Online analytical processing features added** Several OLAP features have been added to the allowed SQL language:

- **ROLLUP operation** For queries with a GROUP BY clause, the ROLLUP operation adds subtotal rows into the result set. Each subtotal row provides an aggregate over a set of rows in the GROUP BY result set.

For more information, see [“Using ROLLUP” \[SQL Anywhere SQL User's Guide, page 408\]](#)

- **The LIST function can include ordered lists** The LIST function has been extended to provide sorted lists of items.

For more information, see [“LIST function \[Aggregate\]” \[SQL Anywhere Reference, page 174\]](#).

- **Additional aggregate functions** Functions have been added to compute sample-based and population-based standard deviations and variances.

For more information, see [“Aggregate functions” \[SQL Anywhere Reference, page 90\]](#).

- ◆ **The CREATE INDEX statement permits an index to be created on a built-in function** This feature is a convenience method that adds a new computed column to a table, and creates an index on that column.

For more information, see [“CREATE INDEX statement” \[SQL Anywhere Reference, page 382\]](#), and [“Creating indexes” \[SQL Anywhere SQL User's Guide, page 87\]](#).

- ◆ **ORDER BY clause allowed in all contexts** In previous releases, many SELECT statements in view definitions, in subqueries, or in UNION operations were not allowed to use an ORDER BY clause. This restriction has now been removed.

In some cases, particularly when combined with the FIRST or TOP clause, using a SELECT with an ORDER BY clause does affect the results of a view definition or a set operation. In other contexts, the ORDER BY clause is allowed but makes no difference to the operation.

- ◆ **SELECT statements can now include START AT as part of the TOP clause** START AT provides additional flexibility in queries that explicitly limit the result set.

For more information, see [“SELECT statement” \[SQL Anywhere Reference, page 623\]](#).

- ◆ **Constraints can now be named** Check constraints, unique constraints, and referential integrity constraints can now be assigned names. This permits modification of table and column constraints by changing individual constraints, rather than by modifying an entire table constraint.

To gain the benefits of this enhancement on databases created before this release, you must upgrade the database file format by unloading and reloading the database.

For more information, see “[ALTER TABLE statement](#)” [*SQL Anywhere Reference*, page 303], “[CREATE TABLE statement](#)” [*SQL Anywhere Reference*, page 427], and “[Using table and column constraints](#)” [*SQL Anywhere SQL User’s Guide*, page 110].

- ◆ **Lateral derived tables permit outer references in the FROM clause** Outer references can now be made from derived tables and from stored procedures in the FROM clause. To indicate that an outer reference is being made, the LATERAL keyword is used.

For more information, see “[FROM clause](#)” [*SQL Anywhere Reference*, page 512].

- ◆ **EXECUTE IMMEDIATE allows more flexible escape character processing** A new option WITH ESCAPES OFF allows escape character processing to be suppressed. This feature makes it easier to construct dynamic statements that include file paths.

For more information, see “[EXECUTE IMMEDIATE statement \[SP\]](#)” [*SQL Anywhere Reference*, page 496].

- ◆ **EXECUTE IMMEDIATE supports queries that return result sets** This new feature allows more dynamic construction of statements inside stored procedures.

☞ For more information, see “[Using the EXECUTE IMMEDIATE statement in procedures](#)” [*SQL Anywhere SQL User’s Guide*, page 806], and “[EXECUTE IMMEDIATE statement \[SP\]](#)” [*SQL Anywhere Reference*, page 496].

- ◆ **CREATE FUNCTION and ALTER FUNCTION now permit Transact-SQL syntax** You can now create user-defined functions in the Transact-SQL dialect that return a scalar value to the calling environment.

☞ For more information, see “[CREATE FUNCTION statement](#)” [*SQL Anywhere Reference*, page 376].

- ◆ **Values of autoincrement columns are now available when inserting multiple rows** When inserting rows through a value-sensitive (keyset

driven) cursor, the newly inserted rows appear at the end of the cursor result set.

A consequence of this change is that the value of an autoincrement column for the most recent row inserted can be found by selecting the last row in the cursor. For example, in embedded SQL the value could be obtained using `FETCH ABSOLUTE -1 cursor-name`.

☞ For more information, see [“Modifying rows through a cursor”](#) [*SQL Anywhere Programming Guide*, page 32].

- ◆ **Remote Data Access now handles UUID/GUID columns** Remote Data Access can now manage SQL Server unique identifier columns.
For more information, see [“Data type conversions: Microsoft SQL Server”](#) [*SQL Anywhere SQL User's Guide*, page 749], and [“UNIQUEIDENTIFIERSTR data type”](#) [*SQL Anywhere Reference*, page 54].
- ◆ **Remote Data Access now names remote connections** Remote Data Access connections made via ODBC are now given names, so that they can be dropped.
☞ For more information, see [“Managing remote data access connections”](#) [*SQL Anywhere SQL User's Guide*, page 736].
- ◆ **New function returns data type of an expression** The `EXPRTYPE` function returns the data type of an expression.
For more information, see [“EXPRTYPE function \[Miscellaneous\]”](#) [*SQL Anywhere Reference*, page 154].
- ◆ **EXIT statement enhanced** The Interactive SQL `EXIT` statement can now set an exit code for Interactive SQL.
For more information, see [“EXIT statement \[Interactive SQL\]”](#) [*SQL Anywhere Reference*, page 499].
- ◆ **OUTPUT statement accepts ASIS keyword** When `ASIS` is specified, values are written to the file without any escaping.
For more information, see [“OUTPUT statement \[Interactive SQL\]”](#) [*SQL Anywhere Reference*, page 577].
- ◆ **Indexes and foreign keys can be altered** The `ALTER INDEX` statement allows indexes and foreign keys to be renamed. It also allows an index type to be changed to clustered or nonclustered for user-created indexes as well as primary or foreign key indexes.
To gain the benefits of clustered indexes on databases created before this release, you must upgrade the database file format by unloading and reloading the database.

For more information, see [“ALTER INDEX statement” \[SQL Anywhere Reference, page 281\]](#).

- ◆ **Multiple distinct aggregates permitted in queries** Aggregate functions can take `DISTINCT column-name` as an argument. In previous versions of the software, only one aggregate function with a `DISTINCT` argument could be included in a query. Now, multiple such functions can be used. The following query is permitted in version 9, but not in earlier versions of the software:

```
SELECT count( DISTINCT first_name ),  
       count( DISTINCT last_name )  
FROM contact
```

- ◆ **Full length and abbreviated day names are recognized in all supported languages for event schedules** When creating events, the database server recognizes both full-length and abbreviated day names in any of the languages supported by Adaptive Server Anywhere. Previously, schedules in non-English languages required full day names. For more information, see [“CREATE EVENT statement” \[SQL Anywhere Reference, page 365\]](#).

- ◆ **Hide procedure text to keep your logic confidential** You can obscure the logic contained in stored procedures, functions, triggers and views using the `SET HIDDEN` option. This allows applications and databases to be distributed without revealing the logic in stored procedures, functions, triggers, and views.

To gain the benefits of this enhancement on databases created before this release, you must upgrade the database using the Upgrade utility.

☞ For more information, see [“Hiding the contents of procedures, functions, triggers and views” \[SQL Anywhere SQL User’s Guide, page 818\]](#).

Administration and
scalability enhancements






- ◆ **The Validation utility gives more detailed return codes** The Validation utility (`dbvalid`) gives more specific return codes to indicate the reason a failure occurs.

☞ For more information, see [“Validation utility \(dbvalid\)” \[SQL Anywhere Database Administration Guide, page 658\]](#).

- ◆ **Two new server properties** Two new server properties have been added. `CommandLine` gives you the line that was used to start the server, and `CompactPlatformVer` gives a condensed version of the `PlatformVer` server property.

☞ For more information, see [“Server-level properties” \[SQL Anywhere Database Administration Guide, page 481\]](#).

- ◆ **New `sp_remote_primary_keys` stored procedure** In order to obtain primary key information about remote tables using remote data access, a new stored procedure called `sp_remote_primary_keys` has been added.
To gain the benefits of this enhancement on databases created before this release, you must upgrade the database using the Upgrade utility.
☞ For more information, see “[sp_remote_primary_keys system procedure](#)” [*SQL Anywhere Reference*, page 931].
- ◆ **New `connection_property` returns the name of the communication link for the connection** The new `CommNetworkLink` connection property returns the name of the communication link for the connection.
☞ For more information, see “[Connection-level properties](#)” [*SQL Anywhere Database Administration Guide*, page 458].
- ◆ **NetWare now supports full character set conversion** In 8.x, NetWare supported single-byte-to-single-byte character set conversion, but in 9.0, all character sets supported by the other platforms are also supported on NetWare.
- ◆ **Unload utility can unload column lists** The Unload utility (`dbunload`) can now unload the column list for the `LOAD TABLE` statements that it generates in the `reload.sql` file, facilitating easier reordering of the columns in a table
☞ For more information, see “[The Unload utility](#)” [*SQL Anywhere Database Administration Guide*, page 642].
- ◆ **Database server registers with LDAP** The database server can now register itself with an LDAP server, so that clients and the Locate Utility (`dblocate`) can query the LDAP server to find it. This allows clients running over a WAN or through a firewall to find servers without specifying the IP address to find such servers. LDAP is only used with TCP/IP, and only on network servers.
☞ For more information, see “[Connecting using an LDAP server](#)” [*SQL Anywhere Database Administration Guide*, page 108] or “[LDAP protocol option \[LDAP\]](#)” [*SQL Anywhere Database Administration Guide*, page 240].
- ◆ **Improved handling of a large number of connections** The liveness timeout value now increases automatically when there are more than 200 connections in an effort to better handle a large number of connections.
☞ For more information, see “[-tl server option](#)” [*SQL Anywhere Database Administration Guide*, page 173] and “[LivenessTimeout connection parameter \[LTO\]](#)” [*SQL Anywhere Database Administration Guide*, page 224].

- ◆ **Request log filtering, host variable support** Output to the request log can now be filtered to include only requests from a specific connection or for a specific database. As well, host variable values can now be output to a request log.
 For more information, see “[sa_server_option system procedure](#)” [*SQL Anywhere Reference*, page 907], “[Monitoring and Improving Performance](#)” [*SQL Anywhere SQL User’s Guide*, page 189], “[sa_get_request_times system procedure](#)” [*SQL Anywhere Reference*, page 863], and “[-zr server option](#)” [*SQL Anywhere Database Administration Guide*, page 186].
- ◆ **BACKUP statement and dbbackup allow renaming of log copy** You can use the BACKUP statement and the Backup utility [dbbackup] to rename the log copy.
 For more information, see “[The Backup utility](#)” [*SQL Anywhere Database Administration Guide*, page 546] and the “[BACKUP statement](#)” [*SQL Anywhere Reference*, page 318].
- ◆ **START DATABASE statement allows log truncation on checkpoint and read-only mode** The START DATABASE statement now allows a database to be started either with log truncation on checkpoint enabled, or in read-only mode.
 For more information, see “[START DATABASE statement](#)” [*SQL Anywhere Reference*, page 650].
- ◆ **Adaptive Server Anywhere supports different auditing options** In previous versions of Adaptive Server Anywhere, you could choose to turn auditing on or off. Now you can specify which options you want to audit.
 For more information, see “[sa_disable_auditing_type](#)” [*SQL Anywhere Reference*, page 855] or “[sa_enable_auditing_type](#)” [*SQL Anywhere Reference*, page 857].
- ◆ **Three new values can be passed to the event_parameter function** Three new values can be passed to the event_parameter function. ScheduleName returns the name of the schedule which fired the event. AppInfo returns the value of the connection_property(‘AppInfo’) for the connection which caused the event. DisconnectReason returns a string indicating why the connection terminated.
 For more information, see “[EVENT_PARAMETER function \[System\]”](#) [*SQL Anywhere Reference*, page 150].
- ◆ **New server property specifies how many concurrent users are connected to the network server** The new LicensesInUse property determines the numbers of concurrent users currently connected to the network server. Each concurrent user is determined by the number of

unique client network addresses connected to the server, not the number of connections. For example, if three client computers are connected to a server, and each client computer has two connections, select property('LicensesInUse') is '3'.

☞ For more information, see “[Server-level properties](#)” [*SQL Anywhere Database Administration Guide*, page 481].

◆ **The Service Creation [dbsvc] utility can now start and stop services**

Two new options have been added to the Service Creation [dbsvc] utility. **Dbsvc -u service_name** starts the service named service_name, and **dbsvc -x service_name** stops the service named service_name.

☞ For more information, see “[Service utility \(dbsvc\)](#)” [*SQL Anywhere Database Administration Guide*, page 616].

◆ **The network server supports the LocalOnly protocol option [LOCAL]**

You can use the LocalOnly protocol option [LOCAL] with the server. Running a server with the LocalOnly protocol option set to YES allows the network server to run as a personal server without experiencing connection or CPU limits.

☞ For more information, see “[LocalOnly protocol option \[LOCAL\]](#)” [*SQL Anywhere Database Administration Guide*, page 241].

◆ **New minimum database server cache size when using Address**

Windowing Extensions The minimum size of the database server cache when using Address Windowing Extensions (AWE) on Windows 2000, Windows XP, and Windows Server 2003 is now 2 MB. In previous releases, the minimum cache size when using AWE was 3 GB-256 MB.

☞ For more information, see “[-cw server option](#)” [*SQL Anywhere Database Administration Guide*, page 137].

◆ **New database property specifies drive type** The new DriveType database property provides information about the drive on which the database file is located.

☞ For more information, see “[Understanding database properties](#)” [*SQL Anywhere Database Administration Guide*, page 458].

◆ **Adaptive Server Anywhere NetWare now faster** The Adaptive Server Anywhere server for NetWare now uses LibC rather than CLIB. LibC is a C runtime library that allows better interaction with the new kernel of the NetWare operating system than the legacy CLIB library. All client-side software for NetWare (including dblib, dbisql, dbconsole, and dbremote) still uses CLIB. This has the benefit of increasing the maximum file size on NetWare to the same as NTFS, allowing multiple CPUs if available,

and allowing TCP and SPX to use Winsock, which is faster than previous versions.

☞ For more information, see [“Physical Limitations” \[SQL Anywhere Database Administration Guide, page 503\]](#) and [“Behavior changes in version 9.0” on page 201](#).

- ◆ **External function enhancements on NetWare** External functions or external stored procedures on NetWare can now use multiple NLMs without naming conflicts.

☞ For more information, see [“External function prototypes” \[SQL Anywhere SQL User’s Guide, page 814\]](#).

- ◆ **Connections can specify language of error messages** Each connection to the database server can now request the language in which the database server reports error messages and various other strings. The language used by the connection is independent of the language used by the server. The database server also uses the language requested by the connection to interpret date strings.

- ◆ **Two new server properties identify processor type** Two new server-level properties have been added. ProcessorArchitecture identifies the processor type, and on platforms where a processor can be emulated NativeProcessorArchitecture identifies the native processor type.

For more information, see [“Server-level properties” \[SQL Anywhere Database Administration Guide, page 481\]](#).

- ◆ **Database password case sensitivity is independent of database case sensitivity** The CREATE DATABASE statement, Initialization [dbinit] utility, and Create Database wizard allow you to specify whether passwords are to be case sensitive or case insensitive. The case sensitivity setting for passwords is independent of the database case sensitivity setting used for string comparisons. The new CaseSensitivePasswords database property allows you to check the password case sensitivity setting for a database.

☞ For more information, see [“CREATE DATABASE statement” \[SQL Anywhere Reference, page 349\]](#) and [“The Initialization utility” \[SQL Anywhere Database Administration Guide, page 573\]](#).

Performance
enhancements (Query
optimization)

The new features listed here are query optimization enhancements that require no user action to use. They take effect without user intervention. If you study query execution plans, you may see the effect of these optimizations.

The optimization enhancements do not require a database upgrade, but they do operate most effectively on a database created using version 9 software.

- ◆ **Cost-based subquery optimization** The optimizer has greatly extended the scope of optimizations that are available for subqueries. In previous releases, subqueries were either rewritten as joins during semantic query optimization or were optimized separately from the remainder of a query. Now subqueries that are too complex to be rewritten as joins can still be optimized as an integral part of the query.
- ◆ **Buffered row fetching improves performance of sequential scans** When reading rows from a database page for a sequential table scan, Adaptive Server Anywhere can now copy rows into a buffer before returning them to the consumer. Depending on the complexity of the query, this can provide significant time savings.
- ◆ **Top N queries executed more efficiently** A new algorithm for executing queries that use the TOP N clause permits faster execution.
 ☞ For more information, see “Sort Top N” [*SQL Anywhere SQL User's Guide*, page 541].
- ◆ **New algorithm for determining which frequencies are kept in histograms** Previously, column histograms created singleton buckets for values with selectivity > 1%. Now, the condition for singleton buckets is relaxed, and instead the histogram tries to keep a minimum number of singleton buckets.
 ☞ For more information, see “Optimizer estimates and column statistics” [*SQL Anywhere SQL User's Guide*, page 506].
- ◆ **Property QueryCachedPlans shows how many query plans are currently cached** The new property, QueryCachedPlans, shows how many query execution plans are currently cached for a given connection, or across all connections. It can be used in combination with QueryCachePages, QueryOptimized, QueryBypassed, and QueryReused to determine the best setting for the max_plans_cached option.
 ☞ For more information, see “Connection-level properties” [*SQL Anywhere Database Administration Guide*, page 458] and “Understanding database properties” [*SQL Anywhere Database Administration Guide*, page 458].
- ◆ **Plans are cached faster for procedure statements** The scope of statements for which access plans are cached has been extended to include queries within stored procedures whose result sets are returned by the procedure to the calling environment. This enhancement eliminates the need to re-optimize some statements.
 ☞ For more information, see “Execution plan caching” [*SQL Anywhere SQL User's Guide*, page 525].

Performance
enhancements (Server
operation)

- ◆ **Index statistics maintained as each index is updated** Statistics are maintained for all indexes, including those on catalog tables, as each index is updated, providing accurate statistics to the optimizer at virtually no performance cost. Statistics persist in SYSATTRIBUTE in the form of one row for each statistic for an index.

To gain the benefits of this enhancement on databases created before this release, you must upgrade the database file format by unloading and reloading the database.

☞ For more information, see “[ISYSATTRIBUTE system table](#)” [[SQL Anywhere Reference](#), page 705].

- ◆ **New performance monitor statistics** Two new performance monitor statistics, Comm: Licenses in Use, and Connection Count, have been added to allow users to track the number of connections in use.

☞ For more information, see “[Communications statistics](#)” [[SQL Anywhere SQL User’s Guide](#), page 244] and “[Miscellaneous statistics](#)” [[SQL Anywhere SQL User’s Guide](#), page 251].

- ◆ **The option APPEND { ON | OFF } has been added to the UNLOAD and UNLOAD TABLE statements** A new APPEND option allows unloaded data to be appended to the end of the specified file.

☞ For more information, see the “[UNLOAD statement](#)” [[SQL Anywhere Reference](#), page 671] or the “[UNLOAD TABLE statement](#)” [[SQL Anywhere Reference](#), page 673].

- ◆ **Temporary tables can now be declared as NOT TRANSACTIONAL** When NOT TRANSACTIONAL is used, the table is not affected by COMMIT or ROLLBACK. This extension is useful when procedures that access the table are called repeatedly without a COMMIT.

☞ For more information, see “[CREATE TABLE statement](#)” [[SQL Anywhere Reference](#), page 427], and “[DECLARE LOCAL TEMPORARY TABLE statement](#)” [[SQL Anywhere Reference](#), page 459].

- ◆ **Persistent index statistics** Maintaining accurate statistics about the physical properties of candidate indexes facilitates the optimizer’s cost based decisions about which indexes to use. Statistics now persist in SYSATTRIBUTE, and are maintained as each index is updated. Additionally, the VALIDATE statement verifies that the statistics on the specified index(es) are accurate and generates an error if they are not. This provides accurate statistics to the optimizer at virtually no performance cost.

To gain the benefits of this enhancement on databases created before this release, you must upgrade the database file format by unloading and reloading the database.

- ☞ For more information, see “[ISYSATTRIBUTE system table](#)” [*SQL Anywhere Reference*, page 705] and the “[VALIDATE INDEX statement](#)” [*SQL Anywhere Reference*, page 690]
- ◆ **New optimistic_wait_for_commit option added** This option is meant to mimic 5.x locking behavior when transactions add foreign rows before primary rows. While it is not intended for general use, it can be helpful when migrating 5.x applications to version 8.x or later.

☞ For more information, see “[wait_for_commit option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 454].
 - ◆ **New extended property function added** The new DB_EXTENDED_PROPERTY function is similar to DB_PROPERTY except that it also allows an optional property-specific string parameter to be specified.

☞ For more information, see “[DB_EXTENDED_PROPERTY function \[System\]](#)” [*SQL Anywhere Reference*, page 136].
 - ◆ **Two new properties added** Two new properties have been added: FileSize and FreePages. Each of these properties can take an optional argument which specifies the dbspace for which the property is being requested.

☞ For more information, see “[Understanding database properties](#)” [*SQL Anywhere Database Administration Guide*, page 458].
 - ◆ **Server's quiet mode enhanced** The server's quiet mode and error logging switches have been enhanced to allow the server to suppress a variety of messages. Additionally, the -qw option has replaced the -q option, and the -qi option has replaced the -Q option.
 - ◆ **Adaptive Server Anywhere plug-in changes** The Adaptive Server Anywhere plug-in for Sybase Central has been reorganized. Much of the information that was previously available in property sheets, dialog boxes, and folders in the left pane is now available on tabs in the right pane. For example, to view information about a foreign key, you now select the table that has the foreign key in the left pane and then select the Foreign Keys tab in the right pane. In previous versions, there was a separate Foreign Keys folder in the left pane.

Several other changes have been made to the plug-in, including the following:

 - The Table Editor is no longer a separate window. Now you edit tables directly in the right pane of Sybase Central.
 - You can edit stored procedures, functions, triggers, and events in the right pane of Sybase Central or in a separate Code Editor window if you want to have multiple windows open at one time.

- The toolbar buttons now change to include options specific to the object selected.
 - The SQL Statements log and server messages (the same information that appears in the Server Messages window) can now be viewed directly in the Sybase Central main window. To view this information, in Sybase Central choose File ► Design Details. The Design Details pane appears at the bottom of the main Sybase Central window.
 - The Adaptive Server Anywhere plug-in provides several new wizards to guide you through tasks, including creating tables, unique constraints, and web services.
- ◆ **Enhanced clipboard support in the Adaptive Server Anywhere plug-in** Clipboard support has been enhanced in the Adaptive Server Anywhere plug-in so you can copy and paste most objects within Sybase Central into other applications, such as Interactive SQL or a text editor. When you copy objects into other applications, depending on the object you select, either the object name or the SQL for the object appears. For example, if you copy an index in Sybase Central and paste it into a text editor, the CREATE INDEX statement for that index appears.
- ☞ For more information, see [“Copying database objects in Sybase Central” \[SQL Anywhere Database Administration Guide, page 512\]](#).
- ◆ **Debugger changes** The database object debugger that lets you debug both stored procedures and Java classes has been integrated into Sybase Central. The user interface has been redesigned.
- ☞ For more information, see [“Debugging Logic in the Database” \[SQL Anywhere SQL User’s Guide, page 821\]](#).
- ◆ **Sybase Central, Interactive SQL, and the Adaptive Server Anywhere Console utility include an option to automatically check for software updates** Sybase Central, Interactive SQL, and the Adaptive Server Anywhere Console utility can be configured to automatically check for software updates. This option can be set from the Options dialog in Interactive SQL and the Adaptive Server Anywhere Console utility, and can be set from the Help menu in Sybase Central when the Adaptive Server Anywhere plug-in is loaded. In previous releases, you had to go to a web site to obtain this information.
- For more information, see [“Options dialog: Check for Updates tab” \[SQL Anywhere 10 Help, page 230\]](#).
- ◆ **Enhancements made to the Adaptive Server Anywhere Console utility** There have been a number of enhancements to the Adaptive Server Anywhere Console utility, including changes to the interface, support for multiple connections, sorting, and drag and drop.

- ◆ **Fast launching of Sybase Central and Interactive SQL** On Windows, Sybase Central and Interactive SQL include a fast launcher that is designed to reduce application startup time when you start Sybase Central or Interactive SQL. Running Adaptive Server Anywhere 9.0.0 starts two background processes, an instance of *dbisqlg.exe* and an instance of *scjview.exe*, which are the fast launcher processes for Interactive SQL and Sybase Central, respectively. Both of these executables are started when the user logs in.

☞ For more information, see “[Options dialog: General tab](#)” [*SQL Anywhere 10 Help*, page 220].

- ◆ **Syntax highlighting editor in Interactive SQL** You can configure the appearance of syntax typed in the SQL Statements pane of Interactive SQL using the Interactive SQL Options dialog.

☞ For more information, see “[Options dialog: Editor tab](#)” [*SQL Anywhere 10 Help*, page 226].

- ◆ **Printing from Interactive SQL** You can print the contents of the SQL Statements pane and of the graphical plan in Interactive SQL.

☞ For more information, see “[Interactive SQL main window description](#)” [*SQL Anywhere Database Administration Guide*, page 517].

- ◆ **Graphical plan enhancements** The graphical query access plan display has been enhanced in several ways:

- The number of rows that passes from one operator to another is indicated by varying line thickness.
- Slow operations are highlighted by a red border.
- The statistics display has been extended and reorganized.
- You can now print the access plan.

- ◆ **Database utilities accept @filename parameters** All of the database administration utilities except Interactive SQL (*dbisql*), the Language Selection utility (*dblang*), and the Adaptive Server Anywhere Console utility (*dbconsole*) now accept parameters contained within a file using the *@file* syntax. The file name can occur at any point in the configuration line, and parameters contained in the file are inserted at that point. Multiple files can be specified, and the file specifier can be used with command line switches. Note that the *@file* syntax is not recursive.

☞ For more information, see the “[@data server option](#)” [*SQL Anywhere Database Administration Guide*, page 127].

- ◆ **Row numbers can appear beside results in Interactive SQL**

Interactive SQL has an option to display row numbers beside results. This option can be set on the Results tab of the Interactive SQL options dialog.

☞ For more information, see [“Options dialog: Results tab” \[SQL Anywhere 10 Help, page 222\]](#).

- ◆ **Interactive SQL can be set as the default editor for .SQL files** On Windows platforms, you can create a file association for .SQL files so that when you double-click the file, Interactive SQL is used to open the file.

☞ For more information, see [“The Interactive SQL utility” \[SQL Anywhere Database Administration Guide, page 582\]](#).

- ◆ **Interactive SQL Command History dialog enhancements** You can now copy and delete commands from the Command History dialog in Interactive SQL, as well as select multiple commands in the window. The command history now persists between Interactive SQL sessions.

☞ For more information, see [“Printing SQL statements” \[SQL Anywhere Database Administration Guide, page 524\]](#).

- ◆ **Warning messages now have W prefix** Prior to version 9.0, all warning and error messages had a prefix of I or E. Warning messages now have a prefix of W. This change affects dbmlsrv9, dbmlsync, dbremote, ssremote, dbltm, and ssqueue.

MobiLink new features

Following is a list of changes and additions to the software introduced in version 9.0.0.

- ◆ **Server-initiated synchronization** Server-initiated synchronization allows you to initiate MobiLink synchronization from the consolidated database. This means you can push data updates to remote databases. The MobiLink component (the Notifier) provides programmable options for determining what changes in the consolidated database will initiate synchronization and how remotes are chosen to receive update messages. The remote component (the Listener) determines how remotes respond.

☞ For more information, see [“Introducing Server-Initiated Synchronization” \[MobiLink Server-Initiated Synchronization User’s Guide, page 1\]](#).

- ◆ **File-based downloads** Downloads can now be processed as a file that can be distributed in any way that files are distributed, such as email, ftp, disk, or multicast file distribution. For this release, this feature can be used only with Adaptive Server Anywhere remote databases.

☞ For more information, see [“MobiLink File-Based Download” \[MobiLink Administration Guide, page 227\]](#).

◆ **New connection scripts `begin_publication` and `end_publication`**

Two new scripts have been added. One of their uses is implementing file-based downloads.

☞ For more information, see “[begin_publication connection event](#)” [*MobiLink Administration Guide*, page 321] and “[end_publication connection event](#)” [*MobiLink Administration Guide*, page 370].

◆ **New connection script `authenticate_parameters`** A new script has been added that allows custom authentication. The new script is invoked during authentication, before the `begin_synchronization` script.

☞ For more information, see “[authenticate_parameters connection event](#)” [*MobiLink Administration Guide*, page 296].

◆ **New option removes blank padding of strings** For columns of type VARCHAR or LONG VARCHAR, the `dbmlsrv9 -b` option removes trailing blanks from strings during synchronization.

☞ For more information, see “[-b option](#)” [*MobiLink Administration Guide*, page 41].

◆ **Option starts new log file with `.old` extension** The `dbmlsrv9 -on` option allows you to set a hard limit on the amount of disk space used by the MobiLink server log.

☞ For more information, see “[-on option](#)” [*MobiLink Administration Guide*, page 66].

◆ **Log progress offsets** The MobiLink synchronization server can now report progress offsets, last upload time, and last download time. To obtain this information, use the `dbmlsrv9` options `-vp` or `-v+`.

☞ For more information, see “[-v option](#)” [*MobiLink Administration Guide*, page 83].

◆ **Handling errors and warnings in .NET and Java synchronization logic** You can now add logic to deal with errors and warnings at the MobiLink synchronization server.

☞ For more information, see “[Handling MobiLink server errors in Java](#)” [*MobiLink Administration Guide*, page 481] and “[Handling MobiLink server errors with .NET](#)” [*MobiLink Administration Guide*, page 527].

◆ **Additions to MobiLink system tables** Two new columns have been added to both the `ml_user` and `ml_subscription` tables. They are `last_upload_time` and `last_download_time`. The default is NOT NULL with a default time of January 1, 1900 00:00:00.

In addition, a `subscription_id` column has been added to `ml_subscription`. The `publication_name` column now contains the publication name.

Enhancements for
Adaptive Server
Anywhere clients

☞ For more information, see “[ml_user](#)” [*MobiLink Administration Guide*, page 631] and “[ml_subscription](#)” [*MobiLink Administration Guide*, page 627].

- ◆ **Upload-only synchronization** You can now choose to perform an upload-only synchronization.

☞ For more information, see the dbmlsync “[-uo option](#)” [*MobiLink Clients*, page 189].

- ◆ **Download-only synchronization** You can now choose to perform a download-only synchronization.

☞ For more information, see “[-ds option](#)” [*MobiLink Clients*, page 125] and “[DownloadOnly \(ds\) extended option](#)” [*MobiLink Clients*, page 137].

- ◆ **Window messages can initiate synchronization** You can now wake up dbmlsync and perform a synchronization by registering a window message as dbas_synchronize and sending it to the dbmlsync top level window.

- ◆ **Load dlls on startup (for Windows CE)** The new dbmlsync option -pd specifies DLLs that should be loaded on startup. This option should be used by everyone using dbmlsync on Windows CE.

☞ For more information, see “[-pd option](#)” [*MobiLink Clients*, page 180].

- ◆ **New way to upgrade or revise schema** The hook `sp_hook_dbmlsync_schema_upgrade` stored procedure has been added to replace the dbmlsync option -i and extended option SiteScriptName (sn).

☞ For more information, see “[sp_hook_dbmlsync_schema_upgrade](#)” [*MobiLink Clients*, page 260].

- ◆ **MobiLink exit codes** To help you track and log the success and failure of your synchronizations, especially when you have multiple synchronizations in a dbmlsync session, there is a new client event hook procedure, `sp_hook_dbmlsync_process_exit_code`. In addition, a new value, exit code, is set in the `#hook_dict` table for the `sp_hook_dbmlsync_abort` hook.

☞ For more information, see “[sp_hook_dbmlsync_process_exit_code](#)” [*MobiLink Clients*, page 258] and “[sp_hook_dbmlsync_abort](#)” [*MobiLink Clients*, page 208].

- ◆ **Enhancements to scheduling** When scheduling is specified, you can reduce the amount of time spent scanning the log by using the new extended option `HoverRescanThreshold (hrt)` or the new hook `sp_hook_dbmlsync_log_rescan`.

☞ For more information, see “[HoverRescanThreshold \(hrt\) extended option](#)” [*MobiLink Clients*, page 141] and “[sp_hook_dbmlsync_log_rescan](#)” [*MobiLink Clients*, page 244].

Languages other than English now support the use of abbreviated day names in schedules. Previously, schedules in non-English languages required full day names.

Two new keywords have been added to scheduling syntax: **INFINITE** instructs dbmlsync to wait indefinitely to be signalled for the next synchronization, and **0** as a day of the month specifies the last day of the month.

☞ For more information, see “[Schedule \(sch\) extended option](#)” [*MobiLink Clients*, page 151].

Enhancements for UltraLite clients

- ◆ **Additional troubleshooting assistance for HotSync conduit** You can configure the HotSync conduit to record troubleshooting information in the HotSync log.

☞ For more information, see “[HotSync log files](#)” [*MobiLink Clients*, page 369].

Performance and monitoring enhancements

- ◆ **Better dbmlsync performance when there are no schema changes** Dbmlsync no longer loads schema information before every synchronization by default. This typically speeds up synchronization on slower handheld devices by 20 seconds.
- ☞ For more information, see “[-sc option](#)” [*MobiLink Clients*, page 185].
- ◆ **Better dbmlsync performance on Windows CE** Dbmlsync no longer uses dbtool9.dll on Windows CE. This means that it uses less memory.
- ◆ **MobiLink Monitor command line options** The MobiLink Monitor can now be started from the command line with a variety of options.
- ☞ For more information, see “[Starting the MobiLink Monitor](#)” [*MobiLink Administration Guide*, page 163].
- ◆ **Enhancements to Redirector** A new parameter, LOG_LEVEL, has been added to allow you to control the verbosity level.
- ☞ For more information, see “[Configuring Redirector properties \(all versions\)](#)” [*MobiLink Administration Guide*, page 187].
- ◆ **Improved liveness** When connecting over TCP/IP, dropped connections are detected more quickly. This frees up MobiLink worker threads more quickly when a connection is dropped, improving throughput.

Miscellaneous

- ◆ **Warning messages now have W prefix** Prior to version 9.0, all warning and error messages had a prefix of I or E. Warning messages now have a prefix of W. This change affects dbmlsrv9, dbmlsync, dbremote, ssremote, dbltm, and ssqueue.

SQL Remote new features

SQL Remote version 9.0.0 includes the following new features.

- ◆ **Warning messages now have W prefix** Prior to version 9.0, all warning and error messages had a prefix of I or E. Warning messages now have a prefix of W. This change affects dbmlsrv9, dbmlsync, dbremote, ssremote, dbltm, and ssqueue.

UltraLite new features

UltraLite development is possible using two kinds of programming interface:

- ◆ **UltraLite components** UltraLite components bring UltraLite database and synchronization features to users of rapid application development tools. They provide a familiar interface for each supported development tool. UltraLite components provide a simple table-based data access interface and also dynamic SQL for more complex queries.
The UltraLite components were introduced in version 8.0.2.
- ◆ **Static development models** Embedded SQL, the static C++ API, and the static Java API are still available. These are now referred to in the documentation as static interfaces to distinguish them from the components.

In particular, note the following:

- ◆ **Native UltraLite for Java** is an UltraLite component, which uses a C/C++ UltraLite runtime. The UltraLite **static Java API** is a pure Java solution, available in previous releases, in which queries must be specified at compile time.
- ◆ **UltraLite for C++** is a component interface. The UltraLite **static C++ API** is a static interface, available in previous releases, in which queries must be specified at compile time.
- ◆ **Embedded SQL** is a static interface, in which queries must be specified at compile time.

Following is a list of changes and additions to the software introduced in version 9.0.

- ◆ **New components** In addition to the components for AppForge MobileVB, eMbedded Visual Basic, and Java, the following components have been introduced:
 - **UltraLite .NET** A component for development using the Visual Studio .NET environment. Applications built with this component can be deployed to devices that support the .NET Compact Framework (version 1.05.0000 or later).
 - ☞ For more information, see [“Introduction to UltraLite.NET” \[UltraLite.NET User’s Guide, page 1\]](#).
 - **C++ component** A component for development using C++ compilers.
 - ☞ For more information, see [“Introduction to UltraLite for C/C++ Developers” \[UltraLite C/C++ User’s Guide, page 3\]](#).
- ◆ **Pocket IE support** The eMbedded Visual Basic component has been upgraded to an ActiveX component. Support has been added for development using JScript, for applications that run from Pocket IE on Windows CE devices.
- ◆ **Dynamic SQL** In addition to the table-based data access interface provided in version 8.0.2, the UltraLite components can now use Dynamic SQL for more complex queries, including multi-table joins.
- ◆ **Connection parameters** Connection parameters for the UltraLite components (except C++) are now exposed as individual properties rather than as a single string. This design makes debugging connection issues easier and makes connection management more straightforward.
 - ☞ For more information, see [“UltraLite Connection String Parameters Reference” \[UltraLite Database User’s Guide, page 117\]](#).
- ◆ **Drag and drop MobileVB component** The MobileVB component can now be dragged on to a form. The properties of the component can be set in the design environment as well as in code.
 - ☞ For more information, see [“UltraLite for MobileVB architecture” \[UltraLite for Crossfire User’s Guide, page 3\]](#).
- ◆ **Multi-process access** The C++ component supports access from more than one process. To develop an application using this model, a separate UltraLite database engine and the application must be linked against a different UltraLite runtime library.
 - ☞ For more information, see [“Compiling and linking your application” \[UltraLite C/C++ User’s Guide, page 33\]](#).

- ◆ **Concurrent synchronization** In previous releases, all access to data was prevented during synchronization. Full access to the data is now provided during the download phase of synchronization. Read-only access is provided during the upload phase.

☞ For more information, see [“Understanding concurrency in UltraLite” \[UltraLite Database User’s Guide, page 19\]](#).




- ◆ **Palm OS enhancements** On the Palm OS the structure of the UltraLite code has been reorganized to make better use of Palm database segments.
- ◆ **Extended error information** More error information is available to applications built using the UltraLite components.
- ◆ **Unicode library available on Windows NT/2000/XP** A Unicode version of the UltraLite runtime library is provided for embedded SQL and static C++ API applications. This version is used by the UltraLite components. When using this library, UltraLite database files are compatible between Windows CE and desktop operating systems.
- ◆ **Windows XP supported as a deployment platform** UltraLite application deployment is now supported on Windows XP.

Behavior changes in version 9.0

This section lists the behavior changes introduced in components of SQL Anywhere Studio version 9.0.

Adaptive Server Anywhere behavior changes

The following is a list of behavior changes from previous versions of the software.

- ◆ **Java objects in the database not supported** Support has been removed for storing data as Java objects. Support is maintained for Java stored procedures.
 For a description of the current support for Java in the database, see [“Introduction to Java in the Database”](#) [*SQL Anywhere Programming Guide*, page 73].
- ◆ **New Greek collation for Windows environment** Greek collations for OEM/DOS character sets existed in previous versions, however, a new Greek collation, 1253ELL, has been added for Windows. When creating a new database in a Greek Windows environment, 1253ELL will be selected automatically if a collation is not specified.
 For more information, see [“Supplied and recommended collations”](#) [*SQL Anywhere Database Administration Guide*, page 301].
- ◆ **New connection limit** The database server now allows one extra DBA connection above the connection limit, to allow a DBA to connect and drop other connections in case of an intentional or accidental denial-of-service.
 For more information, see [“-gm server option”](#) [*SQL Anywhere Database Administration Guide*, page 151].
- ◆ **Personal database server limited to a single processor** In previous versions of the software, the personal database server used a maximum of two CPUs for request processing. Now, the personal server is limited to a single processor.
- ◆ **References to table expressions preceding in the FROM clause may now be used in ON clauses of nested outer joins.** In previous releases, outer references in the ON phrase were permitted. Such outer references must now be indicated by use of the LATERAL keyword. The restriction enforces clarity and conforms to the SQL/99 standard.
 The following query is an example of one that is no longer valid, as it contains an outer reference (highlighted) without use of the LATERAL keyword:

```
SELECT *
FROM T1,
      T2 LEFT OUTER JOIN
          ( T3 LEFT OUTER JOIN T4 ON T1.col1 =
            T2.col2 )
ON T1.col2 = T2.col2
```

For more information, see “[FROM clause](#)” [*SQL Anywhere Reference*, page 512].

- ◆ **Unqualified table references with multiple matches are reported as syntax errors** In previous releases, if a query contained a reference to a table without an owner name specified (an unqualified table reference) and if more than one match was possible on that table, the first match found was used. Unqualified table references now cause an error. See “[Table name '%1' is ambiguous](#)” [*SQL Anywhere Error Messages*, page 433].
- ◆ **LIKE operator with NULL escape character now evaluates to NULL** LIKE predicates containing a NULL escape character now evaluate to NULL. Previously, a LIKE predicate with a NULL escape character was evaluated as if there were no escape character. The new behavior matches the ISO/ANSI specification.
- ◆ **Properties and statistics removed** The ServerIdleWaits database property, and the TaskSwitch and CurrTaskSwitch connection properties have been removed, along with their corresponding performance monitor statistics: Context Switches, Server Idle Waits/sec, Request Queue Waits/sec.
- ◆ **Column statistics are updated on INSERT/UPDATE/DELETE** Statistics are now updated when executing an INSERT, UPDATE, or DELETE statement results in changing a significant amount of data.
- ◆ **Statistics no longer updated during recovery** The server no longer updates statistics during recovery or when executing simple DELETE and UPDATE statements. Simple statements are those that are not optimized and executed directly by the server.
- ◆ **Histogram ranges displayed as the correct data type** The `sa_get_histogram()` system procedure and the histogram [dbhist] utility previously displayed outputted ranges in hash values. Now, outputted histogram ranges match the data in the corresponding column, and are displayed as the correct data type.

☞ For more information, see “[The Histogram utility](#)” [*SQL Anywhere Database Administration Guide*, page 568] and the “[sa_get_histogram system procedure](#)” [*SQL Anywhere Reference*, page 860].

- ◆ **Only one consolidated user permitted per remote database** It is no longer possible to define multiple consolidated users on the same remote database.
 ☞ For more information, see [“GRANT CONSOLIDATE statement \[SQL Remote\]” \[SQL Anywhere Reference, page 528\]](#) or [“REVOKE CONSOLIDATE statement \[SQL Remote\]” \[SQL Anywhere Reference, page 613\]](#).
- ◆ **CommLinks connection parameter uses shared memory if not explicitly specified** Now, connections that do not specify a CommLinks connection parameter always attempt to connect over shared memory.
 ☞ For more information, see [“CommLinks connection parameter \[LINKS\]” \[SQL Anywhere Database Administration Guide, page 205\]](#).
- ◆ **CommLinks connection parameter always attempts shared memory protocol first** When you specify CommLinks=all, Adaptive Server Anywhere always attempts to connect using the shared memory protocol before attempting to connect using other protocols.
 ☞ For more information, see [“CommLinks connection parameter \[LINKS\]” \[SQL Anywhere Database Administration Guide, page 205\]](#).
- ◆ **Connection errors abort process** Previously, connection protocols listed in the CommLinks connection parameter were attempted one by one until a connection occurred. Now, if a connection error occurs during the process, it aborts the connection process immediately, regardless of whether or not all the listed protocols were tried.
 ☞ For more information, see [“CommLinks connection parameter \[LINKS\]” \[SQL Anywhere Database Administration Guide, page 205\]](#).
- ◆ **Default value for prevent_article_pkey_update changed** The default value for the prevent_article_pkey_update database option has been changed to On to reflect the fact that updating primary key values should be avoided. The new default setting disallows primary key updates on primary keys that are part of a publication. You can override this feature by setting the value to OFF.
 ☞ For more information, see [“prevent_article_pkey_update option \[database\] \[MobiLink\]” \[SQL Anywhere Database Administration Guide, page 433\]](#).
- ◆ **Some functions treated as non-deterministic** The RAND, NEWID, and GET_IDENTITY functions are treated as non-deterministic. A consequence is that these functions are not cached during query execution.

☞ For more information, see [“Function caching” \[SQL Anywhere SQL User’s Guide, page 541\]](#).

- ◆ **Performance messages now display database name** The engine performance advice messages now display the database name. This is especially helpful when running more than one database. As well, messages starting with the word *Note* indicate that they are advice messages.
- ◆ **NetWare clients using Adaptive Server Anywhere versions prior to 9.0.0 require upgrade** As a result of enhancements to NetWare support in Adaptive Server Anywhere, NetWare clients using Adaptive Server Anywhere versions prior to 9.0.0 cannot connect to 9.0.0 servers using shared memory unless they have a specific EBF installed. The build numbers are: 7.0.4.3400, 8.0.0.2358, 8.0.1.3088, and 8.0.2.4095. Clients with build numbers before these will simply not find the 9.x server.
- ◆ **Change in syntax for ALTER DATABASE CALIBRATE** The syntax for ALTER DATABASE CALIBRATE TEMPORARY DBSPACE has been changed to ALTER DATABASE CALIBRATE DBSPACE TEMPORARY to make the syntax consistent with other, similar statements.

☞ For more information, see the [“ALTER DATABASE statement” \[SQL Anywhere Reference, page 271\]](#).
- ◆ **Dynamic cache sizing more aggressive** Dynamic cache sizing is now more aggressive at resizing the cache after a new database is started or when a file grows significantly. Prior to this change, statistics were sampled and the cache was resized at most once per minute. Now, after a database is started or a file grows significantly, statistics are sampled and the cache may be resized every five seconds for thirty seconds.

☞ For more information, see [“Using the cache to improve performance” \[SQL Anywhere SQL User’s Guide, page 225\]](#).
- ◆ **Determining the language for interfaces and messages** Two new environment variables, ASLANG and ASCHARSET, control languages used in interfaces (such as Sybase Central or Interactive SQL) and messages. ASLANG specifies the language, and ASCHARSET specifies the character set.

For more information, see the [“SALANG environment variable” \[SQL Anywhere Database Administration Guide, page 268\]](#) or the [“SACHARSET environment variable” \[SQL Anywhere Database Administration Guide, page 267\]](#).

Deprecated and
unsupported features

- ◆ **Rowcount setting now limits the rows returned** The rowcount setting now limits the rows returned by a cursor from the top. It is no longer possible to position to the beginning of the results using an absolute fetch.

You can use the new feature, TOP N / START AT to emulate this behavior if it is needed.

For more information see [“Sort Top N” \[SQL Anywhere SQL User's Guide, page 541\]](#).

Adaptive Server Anywhere -d server option no longer supported As a result of enhancements to NetWare support in Adaptive Server Anywhere, the -d server option is no longer supported.

NetWare 4.x no longer supported As a result of enhancements to NetWare support in Adaptive Server Anywhere, Adaptive Server Anywhere will only run on NetWare version 5.1 SP6 or higher, or version 6.0 SP3 or higher. The correct service packs must be installed or the Adaptive Server Anywhere server will display an error message.

SQLLOCALE environment variable no longer supported SQLLOCALE environment variable has been replaced by two new environment variables, ASLANG and ASCHARSET.

For more information, see [“Behavior changes in version 9.0” on page 201](#).

MobiLink behavior changes

The following is a list of behavior changes from previous versions of the software.

- ◆ **dbmlsync option -i and extended option SiteScriptName are no longer supported** dbmlsync -i and dbmlsync -e sc are no longer supported. They are replaced with a new hook called sp_hook_dbmlsync_schema_upgrade.

☞ For more information, see [“sp_hook_dbmlsync_schema_upgrade” \[MobiLink Clients, page 260\]](#).

- ◆ **Download acknowledgement is now OFF by default** For Adaptive Server Anywhere remotes, the SendDownloadAck extended option now defaults to OFF. For UltraLite remotes, the send_download_ack field of the ul_synch_info struct now defaults to ul_false.

When you upgrade to version 9, you must explicitly set this option On if the application depends on knowing that the remote has applied a download before the commit of the download transaction.

☞ For more information, see [“SendDownloadACK \(sa\) extended option” \[MobiLink Clients, page 154\]](#) and [“Send Download](#)

[Acknowledgment synchronization parameter](#)” [*MobiLink Clients*, page 395].

- ◆ **Some dbmlsync hooks may not work by default on Windows CE devices** The dbmlsync extended option LockTables has been modified to allow you to specify whether tables are locked in shared mode or exclusive mode. The default setting for LockTables, ON, continues to lock tables in shared mode for all platforms other than Windows CE. However, on Windows CE devices, ON now means that tables are locked in exclusive mode. This change provides significant performance enhancements for Windows CE applications.

The dbmlsync event hooks `sp_hook_dbmlsync_download_com_error`, `sp_hook_dbmlsync_download_fatal_sql_error`, and `sp_hook_dbmlsync_download_log_ri_violation` are all executed on separate connections. They will not be able to execute correctly if they attempt to access any synchronization tables that are locked in exclusive mode. If your deployment uses any of these hooks on Windows CE, you may need to set LockTables to SHARE.

☞ For more information, see [“LockTables \(lt\) extended option](#)” [*MobiLink Clients*, page 145], [“sp_hook_dbmlsync_download_com_error \(deprecated\)”](#) [*MobiLink Clients*, page 223], [“sp_hook_dbmlsync_download_fatal_sql_error \(deprecated\)”](#) [*MobiLink Clients*, page 227], and [“sp_hook_dbmlsync_download_log_ri_violation”](#) [*MobiLink Clients*, page 229].

- ◆ **MobiLink synchronization server error codes** The MobiLink synchronization server now provides more information about errors. All MobiLink server error codes are less than -10000, starting at -10001. For dbmlsync, the error appears in the GUI and the output file. For UltraLite, the error is available as a string in the `ul_sync_info` struct.

☞ For more information, see [“MobiLink Synchronization Server Error Messages”](#) [*SQL Anywhere Error Messages*, page 517].

- ◆ **Upload cursors deprecated** The following scripts are deprecated: `upload_cursor`, `new_row_cursor`, and `old_row_cursor`. You should use statement-based scripts for the upload stream.

☞ For more information, see [“Writing scripts to upload rows”](#) [*MobiLink Administration Guide*, page 265].

- ◆ **-zac and -zec deprecated** The MobiLink synchronization server options for generating cursor-based scripts, `-zac` and `-zec`, have been deprecated.

- ◆ **-zd removed** The MobiLink synchronization server option `-zd`, which caused the `last_download` timestamp to be passed last, has been removed. This parameter is now always passed first.

- ◆ **mlxtract deprecated** The mlxtract utility is deprecated.
 ☞ For information, see [“Creating a remote database” \[MobiLink Clients, page 74\]](#).
- ◆ **end_synchronization scripts always called** Prior to version 9.0, the end_synchronization script might not be called if synchronization failed. Now, the script is always called if a begin_synchronization script is called. This means that any cleanup activities you have placed in the end_synchronization script will be performed regardless of whether the synchronization was successful.
 In addition, end_synchronization scripts have a new parameter, sync_ok, that indicates whether the synchronization was successful (1), or failed (0).
 ☞ For more information, see [“end_synchronization connection event” \[MobiLink Administration Guide, page 373\]](#) and [“end_synchronization table event” \[MobiLink Administration Guide, page 375\]](#).
- ◆ **Stream dlls and shared objects renamed** The names of stream dlls and shared objects have been changed to improve consistency with Adaptive Server Anywhere. The following table details the changes:

Old name	New name
dbhttp9	dbmlhttp9
dbhttps9	dbmlhttps9
dbjrsa9	dbmljrsa9
dbjtls9	dbmljtls9
dbrsa9	dbmlrsa9
dbsock9	dbmlsock9
dbtls9	dbmltls9

For more information, see [“Deploying MobiLink Applications” \[MobiLink Administration Guide, page 651\]](#).

- ◆ **ScoutSync no longer supported** ScoutSync is no longer supported.
- ◆ **Schema information no longer reloaded at each synchronization** Prior to version 9.0, dbmlsync reloaded schema information from the database before each synchronization. It now reloads schema information only at dbmlsync startup. You can revert to the old behavior using the dbmlsync -sc option. If you do not use -sc, dbmlsync should be shut down before any schema changes are made to remote databases. Making

schema changes without shutting down dbmlsync could lead to synchronization errors or other unexpected behavior.

☞ For more information, see “-sc option” [*MobiLink Clients*, page 185].

- ◆ **Synchronization now aborts if key scripts are missing** Prior to version 9.0, synchronization would continue even if certain scripts were missing that might result in the loss of data. MobiLink now aborts in this instance. You can use the dbmlsrv9 -fr option to cause an error to be generated instead of failure.

☞ For more information, see “-fr option” [*MobiLink Administration Guide*, page 58].

- ◆ **keep_alive synchronization parameter is removed** The keep_alive synchronization parameter for TCP/IP and HTTP protocols is no longer valid; in effect it is now always set to ON. This was previously the default setting. To control liveness for TCP/IP connections, you can use the liveness_timeout parameter.

☞ For more information, see the liveness_timeout parameter in “CommunicationAddress (adr) extended option” [*MobiLink Clients*, page 127] or “-x option” [*MobiLink Administration Guide*, page 88].

UltraLite behavior changes

The following is a list of behavior changes from previous versions of the software.

- ◆ **Supported platform changes** The following changes have been made to the supported UltraLite deployment platforms.
 - **ScoutSync no longer supported** Support has been dropped for ScoutSync synchronization software.
 - **VxWorks no longer supported** The VxWorks operating system is no longer supported.
 - **JDK 1.1.8 required for pure Java UltraLite** The pure Java static development model requires JDK 1.1.8 or later, rather than JDK 1.1.4 or later.
 - **Palm OS changes** Changes to the UltraLite architecture for Palm OS provides better performance on newer devices. A consequence is that UltraLite requires more dynamic memory than in previous releases. For anything other than very small databases, it is recommended that Palm OS version 3.5 or later be used, with 4 MB or more of memory.

- **MobileBuilder and PRC Tools no longer supported** UltraLite development is no longer supported on the PenRight! MobileBuilder platform. Development using the GNU PRC Tool chain is also no longer supported.
- ◆ **Development platform changes** Application development for UltraLite components is now supported on Windows NT/2000/XP only. Development using the static interfaces is also supported on Windows 98 SE. Other members of the Windows 95/98/Me family are not supported for development purposes.

The supported Metrowerks CodeWarrior versions are now 8 and 9.
- ◆ **Documentation terminology change** The introduction of the UltraLite components requires new names in order to distinguish the different interfaces. The older UltraLite interfaces (embedded SQL, the C++ API, and the Java API) are now named **static interfaces**, as the queries they use must be specified at compile time. The components provide access to **dynamic SQL**.
- ◆ **UltraLite runtime library on Windows NT/2000/XP** The ActiveX and MobileVB components now use a Unicode runtime library on Windows. This runtime library is compatible with version 8.0.2 UltraLite database (.udb) files for Windows, but not with version 8.0.2 UltraLite database files built on other Windows operating systems.
- ◆ **file_name parameter** In previous versions of the software, the file_name parameter used to specify the UltraLite database file name on the desktop would also be used to specify the file name on a device if no platform-specific parameter was supplied. The file_name parameter is now ignored except for on desktop operating systems.
- ◆ **Static Java API changes** The static Java API has changed. The following methods that were on the JdbcDatabase object have been moved to the JdbcConnection object:
 - countUploadRows
 - getLastDownloadTimeDate
 - getLastDownloadTimeLongThe grant and revoke methods have been added to JdbcConnection for use by applications that do not have an explicit JdbcManager object.
- ◆ **Error code changes** Some UltraLite error codes have changed to more specific and useful values. If you test for individual error codes in your application, check the new codes after upgrade.

For example, if you check for SQLE_DATABASE_NOT_FOUND (or the equivalent in one of the UltraLite interfaces) when connecting to a

database, you should change this to
`SQLC_ULTRALITE_DATABASE_NOT_FOUND`.

For a list of error codes, see the SQL error object in the interface you are using.

- ◆ **UL_STORE_PARMS change for embedded SQL** The `UL_STORE_PARMS` macro is now evaluated during the `EXEC SQL CONNECT` statement. The database is no longer started during the `dbinit` call, but rather on connect. This means that `UL_STORE_PARMS` could be evaluated a different number of times if you use multiple connections. It also means that `UL_STORE_PARMS` must be defined before any `EXEC SQL CONNECT` statements.

CHAPTER 5

What's New in Version 8.0.2

About this chapter

This chapter provides an overview of the new features and behavior changes introduced in SQL Anywhere Studio version 8.0.2.

Contents

Topic:	page
New features in version 8.0.2	212
Behavior changes in version 8.0.2	223

New features in version 8.0.2

This section lists the new features introduced in components of SQL Anywhere Studio version 8.0.2.

Adaptive Server Anywhere new features

This section introduces the new features in Adaptive Server Anywhere version 8.0.2. It provides an exhaustive listing of major and minor new features, with cross references to locations where each feature is discussed in detail.

Highlighted new features

- ◆ **Clustered index support** Creating a clustered index on a table causes the rows in that table to be stored in approximately the same order as they appear in the index. You can use the `LOAD TABLE` statement to load a table with information in the clustered order. As you insert information into the table, the clustering characteristics of the table degrade. You can use the `REORGANIZE TABLE` statement to reestablish the clustering order. Clustered indexes can improve performance.

To use clustered indexes on databases created before this release, you must upgrade the database file format by unloading and reloading the database.

☞ For more information, see [“Using clustered indexes” \[SQL Anywhere SQL User’s Guide, page 86\]](#).

- ◆ **Unique identifier support** Adaptive Server Anywhere supports unique identifiers (UUIDs and GUIDs). UUIDs (universally unique identifiers) and GUIDs (globally unique identifiers) are a mechanism for uniquely identifying rows, even across distinct databases in a synchronization environment.

☞ For more information, see [“The NEWID default” \[SQL Anywhere SQL User’s Guide, page 108\]](#).

- ◆ **Update existing rows with ON EXISTING clause** You can use the `ON EXISTING` clause of the `INSERT` statement to update existing rows with new values, as long as the table has a primary key.

☞ For more information, see [“Changing data using INSERT” \[SQL Anywhere SQL User’s Guide, page 486\]](#), or the [“INSERT statement” \[SQL Anywhere Reference, page 548\]](#).

- ◆ **BACKUP statement supported on Windows CE** Adaptive Server Anywhere allows you to create image backups of databases operating on the Windows CE platform, or to rename or truncate the database’s transaction log.

☞ For more information, see “Types of backup” [*SQL Anywhere Database Administration Guide*, page 722], or the “BACKUP statement” [*SQL Anywhere Reference*, page 318].

- ◆ **Graphical plan enhancements** The graphical plan has been enhanced to include more information, resulting in a new look.

☞ For more information, see “Graphical plans” [*SQL Anywhere SQL User's Guide*, page 556].

- ◆ **Use of work tables is now explicit** The use of work tables is now postponed until as late as possible in the plan. When work tables are used, they now appear explicitly in the graphical plan.

☞ For more information, see “Graphical plans” [*SQL Anywhere SQL User's Guide*, page 556] or “Use of work tables in query processing” [*SQL Anywhere SQL User's Guide*, page 235].

- ◆ **New joins added** New joins added to this release include the nested loops semijoin, the nested loops antisemijoin, the hash semijoin and the hash antisemijoin.

☞ For more information, see “Join algorithms” [*SQL Anywhere SQL User's Guide*, page 531].

Function enhancements

- ◆ **Obtain plan for SQL queries of a specific cursor-type** You can now obtain plans for SQL queries based on their cursor type, using the PLAN, EXPLANATION, GRAPHICAL_PLAN functions.

☞ For more information, see “GRAPHICAL_PLAN function [Miscellaneous]” [*SQL Anywhere Reference*, page 157], “EXPLANATION function [Miscellaneous]” [*SQL Anywhere Reference*, page 153], or “PLAN function [Miscellaneous]” [*SQL Anywhere Reference*, page 193].







☞ For information about setting these plan options in Interactive SQL, see “Options dialog: Plan tab” [*SQL Anywhere 10 Help*, page 225].

- ◆ **Character set conversion function** A new function CSCONVERT is available to convert strings between character sets.

☞ For more information, see “CSCONVERT function [String]” [*SQL Anywhere Reference*, page 127].

- ◆ **Variable test function** A new function VAREXISTS is available to test whether a user-defined variable has been created or declared with a given name. After this test, the variable can be created if necessary, and then used safely.

☞ For more information, see “VAREXISTS function [Miscellaneous]” [*SQL Anywhere Reference*, page 247].

- ◆ **Hide procedure text to keep your logic confidential** You can obscure the logic contained in stored procedures, functions, triggers and views using the SET HIDDEN option. This allows applications and databases to be distributed without revealing the logic in stored procedures, functions, triggers, and views.
 For more information, see [“Hiding the contents of procedures, functions, triggers and views” \[SQL Anywhere SQL User’s Guide, page 818\]](#).
- ◆ **LOAD TABLE now accepts delimiters of more than 1 byte** The LOAD TABLE statement now supports delimiters that are up to 255 bytes.
 For more information, see the [“LOAD TABLE statement” \[SQL Anywhere Reference, page 558\]](#).
- ◆ **New statement provides compatibility for Adaptive Server Enterprise and Microsoft SQL Server** You can use the DEALLOCATE statement to release resources associated with a cursor. This statement is provided for Adaptive Server Enterprise and Microsoft SQL Server compatibility.
 For more information, see the [“DEALLOCATE statement” \[SQL Anywhere Reference, page 450\]](#).
- ◆ **ALTER DATABASE statement behaves like dblog utility** You can use the ALTER DATABASE statement to change the transaction log and mirror log names associated with a database file. Previously, you could only do this using the Transaction Log (dblog) utility.
 For more information, see the [“ALTER DATABASE statement” \[SQL Anywhere Reference, page 271\]](#).
- ◆ **LOAD TABLE can be used for both global and local temporary tables** Adaptive Server Anywhere now supports the LOAD TABLE statement on declared local temporary tables. Previously, only global temporary tables were supported.
 For more information, see the [“LOAD TABLE statement” \[SQL Anywhere Reference, page 558\]](#).
- ◆ **SET statement can be used to assign variable values** You can now assign values to variables using the SET statement in Transact-SQL procedures.
- ◆ **INSERT statement now supports WITH AUTO NAME** If you specify WITH AUTO NAME in an INSERT statement, the names of the items in the SELECT list determine the associations of values to destination columns.
 For more information, see [“INSERT statement” \[SQL Anywhere Reference, page 548\]](#).

- ◆ **EXIT statement enhanced** The Interactive SQL EXIT statement can now set an exit code for Interactive SQL.

☞ For more information, see “EXIT statement [Interactive SQL]” [*SQL Anywhere Reference*, page 499].

- ◆ **Specify the optimization goal for a query in the FROM clause** You can use the FASTFIRSTROW table hint to set the optimization goal for the query without setting the optimization_goal option to first-row.

☞ For more information, see “FROM clause” [*SQL Anywhere Reference*, page 512].

Security Enhancements

- ◆ **New utility allows you to hide the contents of files** Configuration files, also known as command files, sometimes contain passwords. As an enhanced security feature, Adaptive Server Anywhere has a new utility, called the File Hiding utility, that allows you to hide the contents of configuration files using simple encryption.

☞ For more information, see “The File Hiding utility” [*SQL Anywhere Database Administration Guide*, page 566]

- ◆ **Certicom encryption changes** Security has been enhanced to support two types of Certicom encryption, ECC_TLS and RSA_TLS. The encryption known in previous versions of Adaptive Server Anywhere as Certicom encryption has been renamed to ECC_TLS encryption. The Certicom parameter is still accepted and is equivalent to ECC_TLS encryption. Adaptive Server Anywhere now also supports RSA_TLS encryption.

☞ For more information, see the “-ec server option” [*SQL Anywhere Database Administration Guide*, page 141] or the “Encryption connection parameter [ENC]” [*SQL Anywhere Database Administration Guide*, page 216].

Performance Enhancements

- ◆ **New connection parameters can improve network responsiveness** The LazyClose and PrefetchOnOpen network connection parameters can improve performance on networks with poor latency or with applications that process many requests.

☞ For information about these parameters, see the “LazyClose connection parameter [LCLOSE]” [*SQL Anywhere Database Administration Guide*, page 224] and the “PrefetchOnOpen connection parameter” [*SQL Anywhere Database Administration Guide*, page 228]

- ◆ **Scattered reads now used on Windows NT/2000/XP** Previously, sequential scans of large tables copied pages to a 64 KB buffer and then into the cache. Now, providing you are running in a Windows NT Service Patch 2 or higher environment, or in a Windows 2000/XP environment,

and provided your page size is at least 4 KB, scattered reads copy the pages directly to the cache, thus saving time and improving performance.

☞ For more information, see “Use an appropriate page size” [[SQL Anywhere SQL User’s Guide, page 218](#)].

- ◆ **Improved time resolution in request logging** The times obtained using procedure profiling or request logging now have a resolution of 1 millisecond. This change primarily affects servers running on Windows operating systems.

- ◆ **Running multiple versions of the Performance Monitor** If you run multiple versions of Adaptive Server Anywhere simultaneously, you can also run multiple versions of the Windows Performance Monitor simultaneously.

☞ For more information about the Windows Performance Monitor, see “Monitoring database statistics from Windows Performance Monitor” [[SQL Anywhere SQL User’s Guide, page 240](#)].

Miscellaneous Enhancements

- ◆ **Changing server’s temp folder via a registry setting** On Windows CE platforms, you can use the registry to specify which temporary directory the server uses.

☞ For more information, see “Registry settings on Windows CE” [[SQL Anywhere Database Administration Guide, page 279](#)].

- ◆ **New iAnywhere JDBC driver** This robust and high-performance JDBC driver enjoys the benefits of ODBC data sources and the Command Sequence client/server protocol. It is an alternative to the jConnect JDBC driver.

☞ For information on the iAnywhere JDBC driver, see “Using the iAnywhere JDBC driver” [[SQL Anywhere Programming Guide, page 337](#)].

☞ For information on choosing a JDBC driver, see “Choosing a JDBC driver” [[SQL Anywhere Programming Guide, page 334](#)].

- ◆ **Triggers can discriminate among the actions that caused a trigger to fire** You can now carry out different actions depending on whether the trigger was fired by an UPDATE, INSERT, or DELETE operation. This feature enables you to share logic among the different events within a single trigger, and yet carry out some actions in an action-dependent manner.

☞ For more information, see “Trigger operation conditions” [[SQL Anywhere Reference, page 27](#)].

- ◆ **return_date_time_as_string is no longer TDS specific** All connections can now use the return_date_time_as_string option.

☞ For more information about this option, see “[return_date_time_as_string option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 437].

- ◆ **Units can be specified when adding space to a dbspace** You can extend database files by a specific size, in units of pages, kilobytes, megabytes, gigabytes, or terabytes.

☞ For more information, see the “[ALTER DBSPACE statement](#)” [*SQL Anywhere Reference*, page 275]

- ◆ **sa_make_object system procedure** This system procedure can be used in a SQL script to ensure that a skeletal instance of an object exists before executing an ALTER statement which provides the actual definition.

☞ For more information, see “[sa_make_object system procedure](#)” [*SQL Anywhere Reference*, page 876].

- ◆ **New global variable compatible with Microsoft SQL Server** A new global variable has been introduced to allow for Microsoft SQL Server compatibility. The @@fetch_status global variable is the same as the @@sqlstatus global variable, except that it returns the status of the most recent fetch in different values.

☞ For more information, see “[Global variables](#)” [*SQL Anywhere Reference*, page 38].

- ◆ **Character set conversion supported on NetWare** NetWare now supports character set conversion.

- ◆ **Information utility reports the version of installed Java classes** The dbinfo utility and a_db_info structure now report the version of the Java classes installed in a database.

☞ For more information, see “[The Information utility](#)” [*SQL Anywhere Database Administration Guide*, page 571] and “[a_db_info structure](#)” [*SQL Anywhere Programming Guide*, page 600].

- ◆ **Suppress warnings on fetch operations** Versions 8.0 and later of the database server return a wider range of fetch warnings than earlier versions of the software. The ODBC Configuration for Adaptive Server Anywhere dialog allows you to suppress warning messages returned from the database server to ensure that they are handled properly for applications that are deployed with earlier versions of the software.

☞ For more information, see “[ODBC Configuration dialog: ODBC tab](#)” [*SQL Anywhere 10 Help*, page 19].

- ◆ **Controlling updates to primary key columns** Setting the new `prevent_article_pkey_update` option to On disallows updates to the primary key columns of tables that are part of a publication. This option helps ensure data integrity, especially in a replication and synchronization environment.

☞ For more information, see the “[prevent_article_pkey_update option \[database\] \[MobiLink\]](#)” [*SQL Anywhere Database Administration Guide*, page 433].

MobiLink new features

Following is a list of changes and additions to the software introduced in version 8.0.2.

- ◆ **Support for .NET** MobiLink now supports Visual Studio .NET programming languages for writing synchronization scripts.

☞ For more information, see “[Writing Synchronization Scripts in .NET](#)” [*MobiLink Administration Guide*, page 517], “[-sl dnet option](#)” [*MobiLink Administration Guide*, page 75], “[ml_add_dnet_connection_script](#)” [*MobiLink Administration Guide*, page 572], and “[ml_add_dnet_table_script](#)” [*MobiLink Administration Guide*, page 573].

- ◆ **Start classes** You can now write Java and .NET code that executes at the time the MobiLink server starts the Java virtual machine or CLR, before the first synchronization.

☞ For more information, see “[User-defined start classes](#)” [*MobiLink Administration Guide*, page 482].

- ◆ **Maintain unique primary keys using UUIDs** A new way to maintain unique primary keys on remote databases is introduced with Universally Unique IDs (UUIDs, also known as GUIDs).

☞ For more information, see “[Using UUIDs](#)” [*MobiLink Administration Guide*, page 120].

- ◆ **New way to handle referential integrity violations** Two new client event hooks, `sp_hook_dbmlsync_download_ri_conflict` and `sp_hook_dbmlsync_download_log_ri_conflict`, are introduced to help you manage referential integrity violations during download.

☞ For more information, see “[sp_hook_dbmlsync_download_ri_violation](#)” [*MobiLink Clients*, page 232] and “[sp_hook_dbmlsync_download_log_ri_violation](#)” [*MobiLink Clients*, page 229].

Performance and
monitoring
enhancements

- ◆ **Simpler way to delete all rows in a remote table** You can now delete all the data in a remote table by including one row in the `download_delete_cursor` that has NULL in every primary-key column.
 ☞ For more information, see [“Writing download_delete_cursor scripts”](#) [*MobiLink Administration Guide*, page 269].
- ◆ **MobiLink Monitor** A graphical tool, the MobiLink Monitor, has been introduced to allow you to see the time taken by every aspect of the synchronization, sorted by MobiLink user or by worker thread.
 ☞ For more information, see [“MobiLink Monitor”](#) [*MobiLink Administration Guide*, page 161].
- ◆ **Users can estimate number of upload rows to dbmlsync** A new `dbmlsync` command line option has been created, `-urc`, which allows you to improve synchronization performance by providing an estimate of the number of rows that will be uploaded.
 ☞ For more information, see [“-urc option”](#) [*MobiLink Clients*, page 190].
- ◆ **Users can specify persistent HTTP/HTTPS connections** You can use the **persistent** option to tell MobiLink to attempt to use the same connection for all HTTP requests in a synchronization. This setting may improve performance. It should only be used when you are connecting directly to MobiLink, and not through an intermediate agent such as a proxy or redirector.
 ☞ For more information, see [“CREATE SYNCHRONIZATION USER statement \[MobiLink\]”](#) [*SQL Anywhere Reference*, page 424].
- ◆ **New ways to control warning messages** Three new `dbmlsrv9` command line options have been created: `-zw`, `-zwd`, and `-zwe`. With `-zw`, you can control which levels of warning message you want reported. With `-zwd`, you can disable specific warning codes. With `-zwe`, you can enable specific that are disabled with `-zw`.
 ☞ For more information, see [“-zw option”](#) [*MobiLink Administration Guide*, page 103], [“-zwd option”](#) [*MobiLink Administration Guide*, page 104] and [“-zwe option”](#) [*MobiLink Administration Guide*, page 105].
- ◆ **New verbose logging options** The `dbmlsync -v` command line option has been altered and expanded. Now, using `-v` alone causes minimum verbosity. To get maximum verbosity, use `-v+`. There are also several new levels that can be specified to fine tune the information that is logged. These options are also available as extended options.
 ☞ For more information, see [“-v option”](#) [*MobiLink Clients*, page 192].

Connection
enhancements

- ◆ **Ping support** The remote database can now ping the MobiLink synchronization server.
 - ☞ For more information, see “[-pi option](#)” [*MobiLink Clients*, page 181] and “[Ping synchronization parameter](#)” [*MobiLink Clients*, page 391].
- ◆ **New synchronization stream** MobiLink now supports the HTTPS protocol. This new stream implements HTTP over SSL/TLS using RSA encryption, and is compatible with any other HTTPS server.
 - ☞ For more information, see “[-x option](#)” [*MobiLink Administration Guide*, page 88] and “[CREATE SYNCHRONIZATION USER statement \[MobiLink\]](#)” [*SQL Anywhere Reference*, page 424].
- ◆ **New buffer_size option** You can now specify a maximum buffer size for a fixed length HTTP message with the **buffer_size** option.
 - ☞ For more information, see “[CREATE SYNCHRONIZATION USER statement \[MobiLink\]](#)” [*SQL Anywhere Reference*, page 424].
- ◆ **Auto-dial for MobiLink clients** MobiLink clients running on Pocket PC 2002 or Windows desktop computers can now connect through dial-up network connections. Using scheduling, your remote can synchronize unattended. The new synchronization stream parameters are **network_name**, **network_connect_timeout**, and **network_leave_open**.
 - ☞ For more information, see “[CREATE SYNCHRONIZATION USER statement \[MobiLink\]](#)” [*SQL Anywhere Reference*, page 424].
- New Web server support ◆ **Servlet Redirector** MobiLink now supports Web servers that support the Java servlet API 2.2, including Apache Tomcat.
 - ☞ For more information, see “[Synchronizing Through a Web Server with the Redirector](#)” [*MobiLink Administration Guide*, page 181].
- Security enhancements ◆ **RSA cipher suite supported** You can now use RSA encryption as well as the existing elliptical-curve encryption for synchronization security. The utilities gencert and readcert support the RSA certificates as well as elliptical-curve certificates.
 - ☞ For more information, see “[MobiLink Transport-Layer Security](#)” [*MobiLink Administration Guide*, page 201].
- ◆ **gencert can sign pregenerated certificate requests** The certificate generation utility gencert has a new command line option that allows you to sign pregenerated certificate requests.
 - ☞ For more information, see “[Certificate generation utility \[gencert\]](#)” [*MobiLink Administration Guide*, page 590].

SQL Remote new features

SQL Remote version 8.0.2 includes the following new features.

- ◆ **Error logs sent to consolidated database** For improved troubleshooting of errors at remote sites, log information can be collected at the consolidated database.

☞ For more information, see “[Troubleshooting errors at remote sites](#)” [*SQL Remote User's Guide*, page 148].

UltraLite new features

UltraLite 8.0.2 introduces several new features:

- ◆ **UltraLite Components** UltraLite database technology can now be used from new development platforms in an easy-to-use fashion. UltraLite Components bring UltraLite technology to users of eMbedded Visual Basic, AppForge MobileVB, and Java. The component for Java is an alternative to the UltraLite for Java described in this book. The component is not a 100% pure Java implementation, but instead uses native classes for better performance.

The UltraLite Component documentation is available in the online books. For a starting point, see *UltraLite Database User's Guide*.

- ◆ **Upgrading UltraLite databases** When deploying a new version of an application, you can now choose to upgrade the schema of UltraLite database to the schema of the new application.

In 9.0.1, ULEnableGenericSchema was replaced by ULRegisterSchemaUpgradeObserver.

- ◆ **Java runtime is thread-safe** The UltraLite Java runtime is now thread-safe, enabling the development of multi-threaded UltraLite applications.
- ◆ **Deleting UltraLite database files** You can delete an UltraLite database file from an application using the ULDropDatabase function.

☞ For more information, see:

- Embedded SQL: “[ULDropDatabase function](#)” [*UltraLite C/C++ User's Guide*, page 290]
- C++ API: Drop Method

- ◆ **Universally unique identifiers** UltraLite databases can now use the UNIQUEIDENTIFIER Adaptive Server Anywhere data type. This type is a BINARY(16) used for storing universally unique identifiers (UUIDs)

or GUIDs). UNIQUEIDENTIFIER columns that use the NEWID function as a default value can guarantee unique primary keys across a whole MobiLink installation, as an alternative to GLOBAL AUTOINCREMENT.

☞ For more information, see “The NEWID default” [[SQL Anywhere SQL User's Guide, page 108](#)].

- ◆ **New security options for synchronization** Two new secure synchronization protocols are introduced in this release. HTTPS is HTTP implemented over a transport-layer security protocol, and RSA is a form of transport-layer security encryption used over HTTP or TCP/IP networks.

These security options use Certicom technology. Use of Certicom technology requires that you obtain the separately-licensable SQL Anywhere Studio security option and is subject to export regulations. For more information on this option, see “SQL Anywhere components” [[Introducing SQL Anywhere 10, page 8](#)].

☞ For more information about HTTPS synchronization, see “Stream Type synchronization parameter” [[MobiLink Clients, page 398](#)].

- ◆ **Reset last download time** To resynchronize previously downloaded data, for example to set an application to a clean state, you can reset the last download timestamp.

☞ For more information, see “ULResetLastDownloadTime function” [[UltraLite C/C++ User's Guide, page 303](#)].

- ◆ **Troubleshooting previous synchronizations** Functions are now available to obtain information about the success or failure of the most recent synchronization. This feature is particularly useful for Palm OS applications that use HotSync, in which case the synchronization is carried out externally to the application.

☞ For more information, see “ULGetSynchResult function” [[UltraLite C/C++ User's Guide, page 293](#)]. This feature is not yet available for UltraLite Java applications.

- ◆ **Generate more and smaller files** The -x option causes the UltraLite generator to write out more and smaller files for C/C++ projects. This option is to help in cases where the generated code is too large for the compiler to handle in a single file.

- ◆ **Improved synchronization observer** The synchronization observer function has been enhanced. More states and fields have been added to the interface to enable the design of more responsive and informative synchronization dialogs.

Behavior changes in version 8.0.2

This section lists the behavior changes introduced in components of SQL Anywhere Studio version 8.0.2.

Adaptive Server Anywhere behavior changes

The following is a list of behavior changes from previous versions of the software.

- ◆ **Windows CE 2.11 no longer supported** Support has been dropped for the Windows CE 2.11 platform.
- ◆ **SH3 and SH4 chips no longer supported** Support for Windows CE devices using the SH3 and SH4 chips has been dropped.
- ◆ **optimization_goal setting** The default setting for the optimization_goal option is set to **All-rows** rather than **first-row**. This affects the execution plan chosen for some queries and so will change performance characteristics.

☞ For more information, see “[optimization_goal option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 425].

- ◆ **xp_cmdshell displays a command window on Windows operating systems** It is now possible to control whether xp_cmdshell starts a new window. The behavior change applies to databases created with or upgraded to version 8.0.2 or later. On older databases, the previous behavior of not displaying a command window is maintained. The new behavior is compatible with other databases such as Adaptive Server Enterprise and Microsoft SQL Server.

You can hide the command window by specifying a second parameter in the call to **xp_cmdshell**.

☞ For more information, see “[xp_cmdshell system procedure](#)” [*SQL Anywhere Reference*, page 935].

- ◆ **Full-length English day names are recognized regardless of the language used by the database server** When creating events, the full-length English day names are recognized by the database server, regardless of the language (German, Chinese, and so on) the database server is using. This means that event definitions in the reload script will be recognized by a server running with a different language.

Events that use the abbreviated English day names (Mon, Tue, and so on) are not recognized by servers running in languages other than English.

☞ For more information, see “[CREATE EVENT statement](#)” [*SQL Anywhere Reference*, page 365].

- ◆ **OPTION settings validated** Integer options with minimum and maximum values are now validated. Setting an option to an invalid value gives the error “Invalid setting for option '%1'” [*SQL Anywhere Error Messages*, page 331].

If you unload and reload a database that contains invalid option settings, they are set to the closest legal value.

The affected options are as follows. The square brackets indicate an inclusive range.

Option	Range
isolation_level	[0, 3]
precision	[0, 127]
scale	[0, 127]
nearest_century	[0, 100]
max_hash_size	[2, 64]
MAX_WORK_TABLE_HASH_SIZE	[2, 64]
first_day_of_week	[1, 7]
default_timestamp_increment	[1, 60000000]

- ◆ **Renamed joins** The names of two joins have changed, both in the graphical plan and in the documentation. Nested loops join not exists (JNE) are now called Nested loops antisemijoin (JNLA), and nested loops exists joins (JE) are now called nested loops semijoins (JNLS)

☞ For more information, see “[Join algorithms](#)” [*SQL Anywhere SQL User's Guide*, page 531].

Deprecated and unsupported features

This list includes features that are no longer supported and that impact existing applications.

- ◆ **-d server option deprecated on Windows** When used on NetWare, the -d option forces the use of POSIX I/O rather than DFS (Direct File System) I/O. In Windows, the option is still allowed on the command line, but is ignored.

MobiLink behavior changes

The following is a list of behavior changes from previous versions of the

software.

- ◆ **serial communications protocol no longer supported** The serial protocol is no longer supported. In its place, you can use HTTP, HTTPS, or TCP/IP.
- ◆ **Certicom no longer a certificate-issuing authority** You can no longer obtain transport-layer security certificates from Certicom. However, you can continue to use the Certicom reqtool utility to generate certificate requests, and you can purchase the certificates from a variety of other sources, including VeriSign and Entrust Technologies.
 ☞ For more information, see <http://www.verisign.com/> or http://www.entrust.com/certificate_services/index.htm.
- ◆ **dbmlsrv option -vw deprecated** The -vw dbmlsrv command line option, which was used to suppress warning messages, has been deprecated. In its place, you can use -zw or -zwd.
 ☞ For more information, see “-zw option” [*MobiLink Administration Guide*, page 103] and “-zwd option” [*MobiLink Administration Guide*, page 104].
- ◆ **dbmlsync option -v behavior change** The -v dbmlsync command line option has been altered and expanded. Now, using -v alone causes minimum verbosity.
 ☞ For more information, see “-v option” [*MobiLink Clients*, page 192].
- ◆ **Full-length English day names are recognized regardless of the language used by the synchronization server** When creating schedules for MobiLink users, publications, and subscriptions, or when specifying scheduling information on the dbmlsync command line, you must use the full-length form of English day names (such as Monday) if you want the schedule to be recognized by a synchronization server running in a language other than English.
 Schedules that use the abbreviated English day names (such as Mon) are not recognized by synchronization servers running in languages other than English.
 ☞ For more information, see “CREATE SYNCHRONIZATION USER statement [MobiLink]” [*SQL Anywhere Reference*, page 424].
- ◆ **Better support for long data in dbmlsync** DBMLSync now handles BLOBs in a much more efficient way while building the upload stream. BLOBs are now read into memory in pieces, so the ability to handle long BLOBs is no longer limited by available memory. When multiple publications are synchronized at one time, BLOB data is stored one time

and shared between the upload streams. The output log now prints the size of the BLOB and its first 32 bytes.

- ◆ **HTTP option use_cookies removed** The use_cookies option has been removed. If you use it, the option is ignored. MobiLink now automatically detects when it needs cookies.

UltraLite behavior changes

The following is a list of behavior changes from previous versions of the software.

- ◆ **Windows CE 2.11 no longer supported** Support has been dropped for the Windows CE 2.11 platform.
- ◆ **SH3 and SH4 chips no longer supported** Support for Windows CE devices using the SH3 and SH4 chips has been dropped.
- ◆ **serial communications protocol no longer supported** The serial protocol is no longer supported. The major use of serial synchronization was from clients on the Palm Computing Platform. These clients can use HotSync synchronization instead.
- ◆ **No transport-layer security on VxWorks** The Certicom libraries that provide transport-layer security for synchronization are no longer supported on the VxWorks operating system.
- ◆ **VxWorks 5.5 not supported** VxWorks 5.3 and 5.4 are the supported versions of the VxWorks operating system.

VxWorks unsupported in version 9

Support for the VxWorks platform is dropped entirely in version 9.

- ◆ **Certicom libraries require JDK 1.2** The Certicom security libraries have been updated with this release. The new libraries for Java applications require JDK 1.2, rather than JDK 1.1.4.

CHAPTER 6

What's New in Version 8.0.1

About this chapter

This chapter provides an overview of the new features and behavior changes introduced in Adaptive Server Anywhere version 8.0.1.

Contents

Topic:	page
New features in version 8.0.1	228
Behavior changes in version 8.0.1	235

New features in version 8.0.1

This section lists the new features introduced in components of SQL Anywhere Studio version 8.0.1.

Adaptive Server Anywhere new features

This section introduces the new features in Adaptive Server Anywhere version 8.0.1. It provides an exhaustive listing of major and minor new features, with cross references to locations where each feature is discussed in detail.

- ◆ **Specify space to be reserved in table pages** You can reduce table fragmentation by specifying the percentage of free space that should be reserved in table pages.

☞ For more information, see “Table fragmentation” [*SQL Anywhere SQL User’s Guide*, page 252] and “ALTER TABLE statement” [*SQL Anywhere Reference*, page 303].

To specify the percentage of space to be allocated on databases created before this release, you must upgrade the database file format by unloading and reloading the database.

- ◆ **New system tables** Two new system tables, SYSATTRIBUTE and SYSATTRIBUTENAME, have been added.

☞ For more information, see “ISYSATTRIBUTE system table” [*SQL Anywhere Reference*, page 705] and “ISYSATTRIBUTENAME system table” [*SQL Anywhere Reference*, page 705].

- ◆ **sa_disk_free_space system procedure** This procedure allows you to determine the space available for your dbspaces, temporary file, transaction log, and transaction log mirror.

☞ For more information, see “sa_disk_free_space system procedure” [*SQL Anywhere Reference*, page 856].

- ◆ **sa_flush_statistics system procedure** Database administrators can use this procedure to ensure that cost model statistics that exist only in the database server cache are flushed out.

☞ For more information, see “sa_flush_statistics system procedure” [*SQL Anywhere Reference*, page 859].

- ◆ **New ways to obtain server message window contents** There is a new system procedure and three new properties that return information from the Server Messages window.

☞ For more information, see “[sa_get_server_messages system procedure](#)” [*SQL Anywhere Reference*, page 864]; and `MessageText`, `MessageTime`, and `MessageWindowSize` in “[Server-level properties](#)” [*SQL Anywhere Database Administration Guide*, page 481].

- ◆ **Determine ANSI equivalency of non-ANSI statements** The `REWRITE` function accepts a new argument, `ANSI`, which causes the function to return the ANSI equivalent of any `SELECT`, `UPDATE`, or `DELETE` statement.

☞ For more information, see “[REWRITE function \[Miscellaneous\]](#)” [*SQL Anywhere Reference*, page 213].

- ◆ **Variable assignment allowed in UPDATE statement** The `SET` clause of the `UPDATE` statement can now be used to assign a value to a variable, in addition to updating the table. This feature is compatible with Adaptive Server Enterprise.

☞ For more information, see “[UPDATE statement](#)” [*SQL Anywhere Reference*, page 676].

- ◆ **Alternative to autoincrement** The `GET_IDENTITY` function is provided as an alternative for allocating identity values to autoincrement columns.

☞ For more information, see “[GET_IDENTITY function \[Miscellaneous\]](#)” [*SQL Anywhere Reference*, page 156].

- ◆ **Square brackets can delimit identifiers** You can use square brackets to delimit identifiers. Square brackets can always be used, regardless of the setting of the `quoted_identifier` option.

☞ For more information, see “[Identifiers](#)” [*SQL Anywhere Reference*, page 8].

- ◆ **Specify isolation level in FROM clause** You can use the `WITH` table-hint argument to specify a locking method for a particular table or view for a particular `SELECT`, `UPDATE`, or `DELETE` statement.

☞ For more information, see “[FROM clause](#)” [*SQL Anywhere Reference*, page 512].

- ◆ **Data Migration wizard** The Data Migration wizard allows you to migrate remote tables to an Adaptive Server Anywhere database from Sybase Central.

You cannot migrate foreign keys if the target database is version 8.0.0 or earlier. To migrate foreign keys, you must upgrade the target database's file format by unloading and reloading the database.

☞ For more information, see “[Migrating databases to SQL Anywhere](#)” [[SQL Anywhere SQL User’s Guide](#), page 695].

- ◆ **Unload a version 5.x or 6.x database from Sybase Central** Sybase Central now allows you to connect to a version 5.x or 6.x database in order to upgrade the database file format using the Unload Database wizard. To do this, you must run the database on a version 8.0.0 or later server.
- ◆ **Back up and shut down your database from the Upgrade Database wizard** You can now back up your database files, including the main database file, the transaction log, and dbspaces from the Sybase Central Upgrade Database wizard. The wizard also allows you to shut down your database when the upgrade is complete.
- ◆ **sa_migrate enhancement** The sa_migrate procedure has an optional argument, *migrate_fkeys* that allows you to specify whether or not you want to migrate foreign key mappings when you migrate tables from a remote database. In previous releases, foreign key mappings were always migrated when you used the sa_migrate procedure.

☞ For more information, see “[sa_migrate system procedure](#)” [[SQL Anywhere Reference](#), page 879].

To use this feature on databases created before this release, you must upgrade the database file format by unloading and reloading the database.

- ◆ **New sort_collation database option** The sort_collation database option allows implicit use of the SORTKEY function on ORDER BY expressions. When the value of this option is set to a valid *collation name* or *collation ID*, any string expression in the ORDER BY clause is treated as if the SORTKEY function had been invoked.

☞ For more information, see “[sort_collation option \[database\]](#)” [[SQL Anywhere Database Administration Guide](#), page 441].

- ◆ **Use an IP address/port to connect to a server** You can use the VerifyServerName=NO protocol option to skip the verification of the server name and allow Adaptive Server Anywhere clients to connect to an Adaptive Server Anywhere server if they know only an IP address/port. The VerifyServerName parameter is only used if DoBroadcast=NONE is specified.

☞ For more information, see “[VerifyServerName protocol option \[VERIFY\]](#)” [[SQL Anywhere Database Administration Guide](#), page 250]

- ◆ **New LocalOnly protocol option controls broadcasts** You can use the LocalOnly protocol option to connect only to a server on the local

computer, if one exists. Setting `LocalOnly=YES` uses the regular broadcast mechanism, except that broadcast responses from servers on other computers are ignored.

☞ For more information, see “[LocalOnly protocol option \[LOCAL\]](#)” [*SQL Anywhere Database Administration Guide*, page 241].

- ◆ **Specify how much of the cache is used for pinning cursors** You can use the `pinned_cursor_percent_of_cache` option to adjust the amount of cache that can be used for pinning cursors. Lowering the limit can improve performance in low memory environments.

☞ For more information, see “[pinned_cursor_percent_of_cache option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 430].

- ◆ **Monitor database file and log file fragmentation** You can use the `DBFileFragments` and `LogFileFragments` database properties to choose monitor file fragmentation. Fragmentation of the transaction log file is usually not a significant concern; however, fragmentation of the database file can be a cause of reduced performance and may warrant use of a disk defragmentation utility.

☞ For more information, see “[Database-level properties](#)” [*SQL Anywhere Database Administration Guide*, page 492].

- ◆ **New connection properties.** Two new connection properties have been added. `LivenessTimeout` returns the liveness timeout of the connection, and `IdleTimeout` returns the idle timeout of the connection.

☞ For more information, see “[Connection-level properties](#)” [*SQL Anywhere Database Administration Guide*, page 458].

- ◆ **New server properties** The new `IdleTimeout` server property returns the default idle timeout value.

☞ For more information, see “[Server-level properties](#)” [*SQL Anywhere Database Administration Guide*, page 481].

- ◆ **Non-deterministic functions** Functions that modify underlying data, or that rely on underlying data that may change during the course of query execution, can be declared `NOT DETERMINISTIC`. Functions that are declared this way are re-evaluated each time they are called during query execution. Otherwise, the function value is cached and re-used for better performance.

☞ For more information, see “[CREATE FUNCTION statement](#)” [*SQL Anywhere Reference*, page 376].

- ◆ **Ensure all transactions in backup are complete** By default, the `BACKUP` statement renames or truncates the transaction log without

waiting for open transactions to complete. You can now ensure that all transactions contained in a backup are complete by specifying a `WAIT AFTER END` clause.

☞ For more information, see [“BACKUP statement” \[SQL Anywhere Reference, page 318\]](#)

MobiLink new features

Following is a list of changes and additions to the software introduced in version 8.0.1.

- ◆ **Full error context reporting** The MobiLink synchronization server now shows the full error context in its output file when an error occurs during synchronization.
☞ For more information, see [“-o option” \[MobiLink Administration Guide, page 65\]](#).
- ◆ **User ID mapping** MobiLink now allows you to more readily find a database user ID or map a MobiLink user name to a user ID.
☞ For more information, see [“modify_user connection event” \[MobiLink Administration Guide, page 414\]](#).
- ◆ **Set address and type as client options** The MobiLink client now allows you specify the communication type and address on the command line to connect to the MobiLink synchronization server.
☞ For more information, see [“dbmlsync extended options” \[MobiLink Clients, page 126\]](#).
- ◆ **Log MobiLink-issued ODBC statements** You can instruct MobiLink to log to an ODBC output file all the ODBC statements issued by MobiLink.
- ◆ **Modify the download timestamp** You can modify the last download timestamp or the next last download timestamp in two new events.
☞ For more information, see [“modify_last_download_timestamp connection event” \[MobiLink Administration Guide, page 408\]](#) and [“modify_next_last_download_timestamp connection event” \[MobiLink Administration Guide, page 411\]](#).
- ◆ **Automatic timestamp conflict tolerance** In the event of a timestamp conflict between the consolidated and remote database, this option allows timestamp values with a precision higher than the lowest-precision to be used for conflict detection purposes.
☞ For more information, see [“-zp option” \[MobiLink Administration Guide, page 98\]](#).

SQL Remote new features

SQL Remote version 8.0.1 includes the following new features.

- ◆ **SMTP user authentication** Parameters are provided for separate user authentication on SMTP servers when using the SMTP/POP message system.

☞ For more information, see “[The SMTP message system](#)” [*SQL Remote User's Guide*, page 139].

UltraLite new features

UltraLite 8.0.1 introduces several new features:

- ◆ **CodeWarrior 8 support** This release supports CodeWarrior version 8.
- ◆ **Support for multi-threaded applications** UltraLite applications can now be multi-threaded on platforms that support this kind of application.
- ◆ **Pocket PC 2002 support** Pocket PC 2002 is added to the list of supported platforms.
- ◆ **JDBC ResultSet methods added** The `ResultSet.findColumn` and `ResultSet.getType` methods are now supported.
- ◆ **Access to information from UltraLite Java** The `JdbcConnection.getLastIdentity` method, `getLastDownloadTime` method, and `JdbcDatabase.countUploadRows` method allow access to useful information. These features were previously available only in C/C++ applications.
- ◆ **User authentication in UltraLite Java** The Java version of UltraLite now supports user authentication.
- ◆ **HotSync synchronization progress displayed** The status field of the HotSync Progress dialog on your desktop computer now shows the progress of synchronization with UltraLite applications.
- ◆ **HotSync configuration** You can configure the HotSync conduit from Palm Desktop.

☞ For more information, see “[Setting the UltraLite HotSync conduit synchronization parameters](#)” [*MobiLink Clients*, page 370].

- ◆ **Automatic scripting from UltraLite applications** UltraLite applications can now provide column names to the MobiLink synchronization server so that synchronization scripts can be automatically generated.

- ◆ **Get SQL data type of a column from the C++ API** The `GetColumnSQLType` method returns the data type of a column.
- ◆ **Optional checkpoint during synchronization** Synchronizations that download large numbers of updates can cause the UltraLite database to grow significantly in size. This growth can be limited by carrying out checkpoints during synchronization. The new **checkpoint_store** synchronization parameter controls checkpointing. By default, no checkpoints are carried out.

☞ For more information, see “[Checkpoint Store synchronization parameter](#)” [*MobiLink Clients*, page 385].

Behavior changes in version 8.0.1

This section lists the behavior changes introduced in components of SQL Anywhere Studio version 8.0.1.



Adaptive Server Anywhere behavior changes

The following is a list of behavior changes from previous versions of the software.

- ◆ **New naming convention for renamed transaction log files** Double digits at the end of transaction log files renamed during backup have been changed to double characters. For example, the renamed log file from the first backup on December 10, 2000, is now named *001210AA.log* instead of *00121001.log*. The first two digits indicate the year, the second two digits indicate the month, the third two digits indicate the day of the month, and the final two characters distinguish among different backups made on the same day. This increases the number of backups possible in a day from 100 to 676.
- ◆ **LOAD TABLE now recalculates computed columns** LOAD TABLE now detects computed columns and evaluates them for each row inserted into the table.
- ◆ **Adaptive Server Anywhere Console utility (dbconsole) now allows connections to be reconnected** Previously an Adaptive Server Anywhere Console utility (dbconsole) session only allowed one connection. Connections can now be disconnected and reconnected without exiting the application.

Deprecated and unsupported features

This list includes features that are no longer supported and that impact existing applications.

- ◆ **DEBUG connection parameter deprecated** The DEBUG connection parameter has been deprecated. You can still use LOG parameter to create a log file containing the debug information. From version 8.0.1 on, LOG=filename does what DEBUG=YES;LOG=filename used to do.
 For more information, see “[Connection parameters](#)” [*SQL Anywhere Database Administration Guide*, page 200].
- ◆ **AGENT connection parameter deprecated** The AGENT connection parameter has been deprecated. You can use the CommLinks parameter with appropriate protocol options to achieve the same behavior.
 For more information, see “[Connection parameters](#)” [*SQL Anywhere Database Administration Guide*, page 200].

- ◆ **Port connection property removed** The port connection parameter has been removed.
- ◆ **Adaptive Server Anywhere Translation Driver removed** Use of translation drivers is no longer recommended. The server automatically handles character set conversion.
- ◆ **SharedMemory tried first** The ports specified in the LINKS= connection parameter were tried in the order in which they were specified. Now, if the sharedmemory (shmem) port is specified, it is tried first, followed by the other ports specified in the order in which they appear.
- ◆ **GLOBAL AUTOINCREMENT** The default value has been changed from 0 to 2147483647. `global_database_id` can now be set to 0 and will cause values to be generated starting at 1.

MobiLink behavior changes

- ◆ **Timestamp mismatch notification** When the timestamps between consolidated and remote databases are at variance, the MobiLink synchronization server will log a warning with each synchronization.
- ◆ **GLOBAL AUTOINCREMENT** The default value has been changed from 0 to 2147483647. `global_database_id` can now be set to 0 and will cause values to be generated starting at 1.

It is still the case that if `global_database_id` is not set, or is set to the default value, attempts to cause a global autoincrement value to be generated result in a NULL. This commonly gives an error when attempting to insert the value into a non-nullable primary key column and is the indication that the `global_database_id` option has not been set.

Disallowing a setting of 0 for `global_database_id` prevented generation of values starting at 1. Instead, values would start at the partition size specified for the column.

☞ For more information, see “[global_database_id option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 399].

- ◆ **dbmlstop performs soft shutdown** By default (if none of -w, -f, -h or -t are specified), `dbmlstop` does a soft shutdown. This means that it stops accepting new connections and exits when the current synchronizations are complete.

☞ For more information, see “[MobiLink stop utility \[mlstop\]](#)” [*MobiLink Administration Guide*, page 584].

UltraLite behavior changes

- ◆ **Palm database backup** In previous releases, if the ULUtil application was used to backup a database, the database would be backed up on each subsequent HotSync operation.

Most UltraLite data is effectively backed up by synchronization. As the most common use of an explicit backup is to create an initial database for deployment, continuing to make backups on HotSync is not the desired behavior in most cases. Now, each time an UltraLite application starts, it disables backups on future HotSync operations.

If you want to explicitly require backups for databases every time a HotSync is performed, you can do so by setting the **palm_all_backup** parameter in the UL_STORE_PARMS macro.

Deprecated and
unsupported features

UltraLite support for synchronization on the Palm Computing Platform using ScoutSync technology is deprecated. Version 8.0.x will continue to support ScoutSync up to version 3.6, but the next major release of SQL Anywhere Studio will not support ScoutSync.

CHAPTER 7

What's New in Version 8.0.0

About this chapter

This chapter provides an overview of the new features and behavior changes introduced in Adaptive Server Anywhere version 8.0.0.

Contents

Topic:	page
New features in version 8	240
Behavior changes in version 8	266

New features in version 8

This section lists the new features introduced in components of SQL Anywhere Studio version 8.

Adaptive Server Anywhere new features

This section introduces the new features in Adaptive Server Anywhere version 8.0. It provides an exhaustive listing of major and minor new features, with cross references to locations where details of each feature appear in the manuals.

If you have the printed version of this book, and if you do not have the complete SQL Anywhere Studio documentation set, you should look in the online documentation for a detailed description of each feature.

Some new features require that you upgrade the database to version 8, or that you upgrade the database file format by unloading and reloading the database. If a database upgrade or file format upgrade is required to access a particular feature, the requirement is indicated in the description below.

☞ For information on how to carry out these tasks, see [“Upgrading to SQL Anywhere 10” on page 333](#).

The Adaptive Server Anywhere new features are grouped under the following headings:

- ◆ [“Query processing and database performance” on page 240](#)
- ◆ [“Security” on page 243](#)
- ◆ [“SQL features” on page 244](#)
- ◆ [“Development and administration tools” on page 245](#)
- ◆ [“Application development” on page 247](#)
- ◆ [“Administration and troubleshooting” on page 248](#)
- ◆ [“Client/server connections” on page 252](#)
- ◆ [“Java in the database” on page 253](#)
- ◆ [“Documentation” on page 254](#)
- ◆ [“Miscellaneous” on page 254](#)

Query processing and database performance

- ◆ **Improved query processing** This version includes enhancements to the query execution engine and the optimizer, resulting in a significant

improvement in performance, especially for complex queries. Enhancements to Adaptive Server Anywhere query processing include the following:

- More sophisticated internal processing of joins.
- Improvements to the optimizer's cost model used to assess alternative access plans.
- Improvements to the execution model.

☞ Most of these changes are internal. Documentation is provided in “Query Optimization and Execution” [*SQL Anywhere SQL User's Guide*, page 489].

An effect of these changes is that it is no longer the case that the materialization of results is necessarily inefficient. Use of temporary work tables may be a very efficient way to execute a query. For more information, see “Use of work tables in query processing” [*SQL Anywhere SQL User's Guide*, page 235].

The optimizer now performs cost-based selection of indexes, and does not solely rely on predicate selectivities as was the case with prior releases.

Much of the improved query processing does not require an upgraded database. To use the new cost model on databases created before this release, you must upgrade the database file format by unloading and reloading the database.

- ◆ **New index type** A new type of index has been added that improves performance for multiple column indexes and for indexes that include wide columns. It is a compressed B-tree index.

Adaptive Server Anywhere automatically creates the appropriate type of index based on index width (the sum of the width of all columns in the index). A compressed B-tree index is created when the width of the index is greater than nine bytes and less than one-eighth of the page size to a maximum of 256 bytes; otherwise, Adaptive Server Anywhere creates hash B-tree indexes.

The WITH HASH SIZE clause of the CREATE INDEX statement is deprecated.

☞ For more information about these indexes, see “Types of index” [*SQL Anywhere SQL User's Guide*, page 573].

To use the new index types on databases created before this release, you must upgrade the database file format by unloading and reloading the database.

A new limitation is imposed: foreign key indexes must have the same size and type as the corresponding primary key index.

dbunload now omits the hash size specification if it was originally specified with the default (WITH HASH SIZE 10).

- ◆ **New database option `optimization_goal`** Determines whether query processing is optimized towards returning the first row quickly, or minimizing the cost of returning the complete result set. The default is to optimize for the first rows.

☞ For more information, see “[optimization_goal option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 425].

- ◆ **Performance enhancements for table scans** Databases created in Adaptive Server Anywhere 8.0 with 2K, 4K, or 8K pages have performance-enhancements for queries that require sequential table scans. Adaptive Server Anywhere creates bitmaps, also known as page maps, for large tables. A bitmap lists all of the pages containing data for a given table. This feature permits searching large tables in only one I/O operation.

☞ For more information, see “[Table and page sizes](#)” [*SQL Anywhere SQL User's Guide*, page 565].

To gain the benefits of this enhancement on databases created before this release, you must upgrade the database file format by unloading and reloading the database.

- ◆ **Improved storage of checkpoint log** The checkpoint log is now stored in consecutive pages at the end of the database file. This leads to improved performance by allowing sequential scans and multipage writes of the material in the checkpoint log.

☞ For more information about the checkpoint log, see “[Checkpoints and the checkpoint log](#)” [*SQL Anywhere Database Administration Guide*, page 738].

To gain the benefits of this enhancement on databases created before this release, you must upgrade the database file format by unloading and reloading the database.

- ◆ **Plan caching** Adaptive Server Anywhere now caches execution plans for queries and INSERT, UPDATE and DELETE statements performed inside stored procedures, user-defined functions, and triggers. The maximum number of plans to cache is specified with the option setting `max_plans_cached`. To disable plan caching, set this option to 0.

☞ For more information, see “[Execution plan caching](#)” [*SQL Anywhere SQL User's Guide*, page 525].

- ◆ **Overriding the default I/O cost model** You can now override the default I/O cost model using the ALTER DATABASE statement with the CALIBRATE clause.
 ☞ For more information, see “ALTER DATABASE statement” [*SQL Anywhere Reference*, page 271].
- ◆ **New database option max_plans_cached** Sets the maximum number of execution plans that are stored in cache.
 ☞ For more information, see “max_plans_cached option [database]” [*SQL Anywhere Database Administration Guide*, page 413].
- ◆ **New database option min_table_size_for_histogram** This option sets the minimum table size for which histograms are created. Histograms store information about the distribution of values in a column, and the optimizer uses them to choose an efficient execution plan.

Security

- ◆ **Strong encryption over TCP/IP** Adaptive Server Anywhere now supports certificate-based encryption over TCP/IP ports on Solaris, Linux, NetWare, and all supported Windows operating systems with the exception of Windows CE. Strong encryption protects the confidentiality and integrity of network packets as they pass between the client and the server. This encryption is also called Transport Layer Security (TLS).
 The database server -ec command line option allows you to set the server's connection parameters and replaces the -e command line option in previous versions of Adaptive Server Anywhere. You can set the client connection parameters with the encryption connection parameter.
 ☞ For more information, see “-ec server option” [*SQL Anywhere Database Administration Guide*, page 141] and “Encryption connection parameter [ENC]” [*SQL Anywhere Database Administration Guide*, page 216].
 To use this feature, you must use version 8 software at both the client and the server. You do not need to upgrade the database.
- ◆ **Strong encryption of the database file** The database file itself can now be strongly encrypted for greater security, especially on notebook and laptop computers prone to theft.
 ☞ For more information, see the following locations:
 - “Initialization utility (dbinit)” [*SQL Anywhere Database Administration Guide*, page 573]
 - “-ek database option” [*SQL Anywhere Database Administration Guide*, page 192]

- “-ep server option” [*SQL Anywhere Database Administration Guide*, page 144]
- “CREATE DECRYPTED FILE statement” [*SQL Anywhere Reference*, page 358].

You must use version 8 software to create encrypted database files.

SQL features

- ◆ **Full outer joins** Full outer joins are now supported. In addition, the keyword OUTER is now optional for right, left, and full outer joins.
☞ For more information, see “Outer joins” [*SQL Anywhere SQL User’s Guide*, page 346].
- ◆ **CASE statements** The ANSI standard allows two forms of CASE statements. Adaptive Server Anywhere 8.0 supports both syntaxes.
☞ For more information, see “CASE statement” [*SQL Anywhere Reference*, page 333].
- ◆ **WAITFOR statement** This statement delays processing for the current connection for a specified amount of time or until a given time.
☞ For more information, see “WAITFOR statement” [*SQL Anywhere Reference*, page 691].
- ◆ **RAISERROR statement allows connections to be disallowed** This statement can now be used to disallow or limit connections.
☞ For more information, see “RAISERROR statement [T-SQL]” [*SQL Anywhere Reference*, page 591].
- ◆ **Timezone adjustment** To permit easier coordination of date/time values across time zones, the following new features have been added:
 - **CURRENT UTC TIMESTAMP** Adjusts the time zone value by the server’s time zone adjustment value.
 - **DEFAULT UTC TIMESTAMP** Specifies a default value for INSERTs and sets updated columns to the value.
 - **TimeZoneAdjustment property** returns the number of minutes that must be added to the Coordinated Universal Time (UTC) to display the new local time.
 - **time_zone_adjustment option** Allows a connection’s time zone adjustment to be modified.
- ◆ **New collation functions** The SORTKEY function generates values that can be used to sort character data. SORTKEY allows you to perform

sorting beyond the default behavior of Adaptive Server Anywhere collation.

The COMPARE function allows you to directly compare two character strings based on alternate collation rules.

☞ For more information, see [“SORTKEY function \[String\]” \[SQL Anywhere Reference, page 223\]](#) and [“COMPARE function \[String\]” \[SQL Anywhere Reference, page 113\]](#).

- ◆ **ERRORMSG function** The new SQL function ERRORMSG can be used to obtain error messages.

☞ For more information, see [“ERRORMSG function \[Miscellaneous\]” \[SQL Anywhere Reference, page 146\]](#).

- ◆ **Data type conversion functions** The ISDATE and ISNUMERIC functions test if a string can be converted to a date or number, respectively.

☞ For more information, see [“ISDATE function \[Data type conversion\]” \[SQL Anywhere Reference, page 169\]](#), and [“ISNUMERIC function \[Miscellaneous\]” \[SQL Anywhere Reference, page 171\]](#)

Development and administration tools

- ◆ **Accessibility features** SQL Anywhere Studio is compliant with Section 508 of the US Federal Rehabilitation Act. The user interfaces and documentation have been prepared in compliance with the act. An accessibility enablement component provides software that enables the use of accessibility tools. The accessibility enablement component is not installed by default.

☞ For more information, see [“SQL Anywhere components” \[Introducing SQL Anywhere 10, page 8\]](#).

- ◆ **Query Editor** A graphical query editor has been added to Interactive SQL. With the Query Editor, you can create or edit SELECT statements without using SQL code. You can open the Query Editor in Interactive SQL by clicking Tools ► Edit Query.

☞ For more information, see [“Introducing the Query Editor” \[SQL Anywhere 10 Help, page 262\]](#).

- ◆ **Editable data in Interactive SQL and Sybase Central** You can update the database by editing Interactive SQL result sets, and by editing tables and views in Sybase Central. You can copy, edit, insert, and delete row values.

Data displayed in Sybase Central can be copied to the clipboard.

☞ For more information, see “Editing result sets in Interactive SQL” [[SQL Anywhere Database Administration Guide, page 524](#)].

◆ **Interactive SQL supports SQL escape syntax handling**

Interactive SQL now supports JDBC escape syntax that allows you to access a library of functions implemented by the JDBC driver.

☞ For more information, see “Using JDBC escape syntax” [[SQL Anywhere Programming Guide, page 359](#)].

◆ **Procedure profiling** Sybase Central contains a Profile tab that displays information about the number of calls and execution times for stored procedures, functions, events, and triggers. You can also view information about the execution speed for each line within a procedure. Profiling information is available through Sybase Central and SQL stored procedures.

☞ For more information about viewing procedure profiling information in Sybase Central, see “Profiling database procedures” [[SQL Anywhere SQL User’s Guide, page 255](#)].

☞ For more information about obtaining procedure profiling information with SQL stored procedures, see “[sa_procedure_profile_summary system procedure](#)” [[SQL Anywhere Reference, page 898](#)] and “[sa_procedure_profile system procedure](#)” [[SQL Anywhere Reference, page 895](#)].

☞ To use this feature, you must upgrade the database.

◆ **Improved information for access plans** There are two new ways to view the plan, a graphical display and a graphical display with statistics. These new plans provide more information about the processing cost of your query, and allow you to examine the cost of subsets of the query. The default access plan is now the graphical plan. The long and short plans are now based on the Ariadne syntax used by Adaptive Server Enterprise, and have new abbreviations.

☞ For more information, see “Reading execution plans” [[SQL Anywhere SQL User’s Guide, page 545](#)].



◆ **Results pane displays query execution plan** The Interactive SQL Results pane now has a Results tab. The Results tab displays the results of your query, and the Plan tab displays the execution plan for the query. Previously, the query execution plan appeared in the Interactive SQL Messages pane.

☞ For more information, see “Options dialog: Results tab” [[SQL Anywhere 10 Help, page 222](#)].

- ◆ **Results pane displays UltraLite plan** The Interactive SQL Results pane now has an UltraLite Plan tab. This tab displays the UltraLite plan optimization strategy in XML format, as a string.
- ◆ **XML export using the OUTPUT statement** You can export query results as XML format. The output has an embedded DTD. Binary values are encoded in CDATA blocks with the binary data rendered as two-hexadecimal-digit strings.
 - ☞ For more information, see “[OUTPUT statement \[Interactive SQL\]](#)” [*SQL Anywhere Reference*, page 577].
- ◆ **Interactive SQL batch options** Additional control is given to Interactive SQL when running batch files, through the -codepage and -onerror command line options. Also, the -d1 command line option provides feedback useful for debugging batch files.
 - ☞ For more information, see “[The Interactive SQL utility](#)” [*SQL Anywhere Database Administration Guide*, page 582].

Application development




- ◆ **New cursor types** The cursors supplied by Adaptive Server Anywhere have been enhanced to provide cleaner semantics, to better match new cursor types such as keyset-driven cursors, and to take advantage of the new query optimization possibilities.
 - ☞ For more information, see “[SQL Anywhere cursors](#)” [*SQL Anywhere Programming Guide*, page 39].
- ◆ **Improved fetching for long columns** The amount of data that can be fetched in a single operation has been increased from 32 KB to a configurable value with a default of 256 KB. In ODBC the value can be set using the SQL_ATTR_MAX_LENGTH statement attribute. In embedded SQL, use the DT_LONGVARCHAR and DT_LONGBINARY types.
 - ☞ For more information, see “[Retrieving data](#)” [*SQL Anywhere Programming Guide*, page 324], and “[Sending and retrieving long values](#)” [*SQL Anywhere Programming Guide*, page 419].
- ◆ **New embedded SQL function to obtain database properties** The function `db_get_property` can be used to obtain database properties.
 - ☞ For more information, see “[db_get_property function](#)” [*SQL Anywhere Programming Guide*, page 441]. For information on database properties, see “[Understanding database properties](#)” [*SQL Anywhere Database Administration Guide*, page 458].

- ◆ **blocking_timeout option** The new `blocking_timeout` option lets you control how long a transaction waits to obtain a lock.
 For more information, see “[blocking_timeout option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 380].
- ◆ **return_date_time_as_string option** The `return_date_time_as_string` option allows you to control how date, time, and timestamp values are returned over jConnect and Open Client.
 For more information, see “[return_date_time_as_string option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 437].

Administration and troubleshooting

In addition to the administration enhancements added to Sybase Central, listed above, version 8 includes the following administration enhancements.

- ◆ **Improve table performance without disrupting access** The `REORGANIZE TABLE` statement can be used to improve performance when a full rebuild of the database is not possible, due to the requirements for continuous access to the database. Use this statement to defragment rows in a table, or to compress indexes which have become sparse due to `DELETEs`. It can also reduce the total number of pages used to store the table and its indexes, as well as reduce the number of levels in an index tree.

To reorganize tables based on a primary key, foreign key, or index, the database must be Adaptive Server Anywhere version 7 or above.
 For more information, see “[REORGANIZE TABLE statement](#)” [*SQL Anywhere Reference*, page 603].
- ◆ **Fast database validation** A new type of validation check has been added that reduces the amount of time it takes to validate a database. This option is of particular interest to people who need to validate large databases with small cache sizes. Affected tools include the `sa_validate` system procedure, the Validation utility (**dbvalid**) and the `VALIDATE TABLE` statement.
 For more information, see “[Improving performance when validating databases](#)” [*SQL Anywhere Database Administration Guide*, page 743].
 To use this feature on databases created before this release, you must upgrade the database file format by unloading and reloading the database.
- ◆ **Backup does not need to wait for outstanding transactions to complete** If a backup instruction requires the transaction log to be truncated or renamed, uncommitted transactions are carried forward to

the new transaction log. This means that the server no longer waits for outstanding transactions to be committed or rolled back before initiating a backup.

☞ For more information, see [“Log Translation utility \(dbtran\)”](#) [*SQL Anywhere Database Administration Guide*, page 596] and [“Backup internals”](#) [*SQL Anywhere Database Administration Guide*, page 737].

To use this feature on databases created before this release, you must upgrade the database file format by unloading and reloading the database.

- ◆ **Obtaining fragmentation statistics** File, table, and index fragmentation can all decrease performance. In Adaptive Server Anywhere 8.0 when you start a database on Windows NT, the server automatically displays information about the number of file fragments in each dbspace.

The new system procedures, `sa_table_fragmentation` and `sa_index_density`, allow database administrators to obtain information about the fragmentation in a database's tables and indexes.

☞ For more information about file fragmentation, see [“File fragmentation”](#) [*SQL Anywhere SQL User's Guide*, page 252].





☞ For more information about table fragmentation, see [“Table fragmentation”](#) [*SQL Anywhere SQL User's Guide*, page 252] and [“sa_table_fragmentation system procedure”](#) [*SQL Anywhere Reference*, page 920].

☞ For more information about index fragmentation, see [“Index fragmentation”](#) [*SQL Anywhere SQL User's Guide*, page 253] and [“sa_index_density system procedure”](#) [*SQL Anywhere Reference*, page 866].

- ◆ **Obtain the most recently prepared SQL statement for a connection** The database server `-zl` command line option turns on capturing of the most recently prepared SQL statement for each connection to databases on a server. You can also turn on this feature using the `sa_server_option` stored procedure with the `remember_last_statement` setting.

When this feature is turned on, the **LastStatement** property function and the `sa_conn_activity` system procedure return the most recently prepared SQL statement for the current connection and all connections to databases on a server respectively.

☞ For more information, see [“-zl server option”](#) [*SQL Anywhere Database Administration Guide*, page 184], [“sa_conn_activity system procedure”](#) [*SQL Anywhere Reference*, page 837], and [“sa_server_option system procedure”](#) [*SQL Anywhere Reference*, page 907].

- ◆ **-cw command line option** This server option lets you use cache sizes up to 64 GB on Windows 2000, Windows XP, and Windows Server 2003.
 For more information, see “-cw server option” [[SQL Anywhere Database Administration Guide, page 137](#)].
- ◆ **-qp option** This server option lets you suppress messages about performance in the Server Messages window.
 For more information, see “-qp server option” [[SQL Anywhere Database Administration Guide, page 165](#)].
- ◆ **Improved debugging server log** The information logged in the connection debugger has been improved to give more context about the portion of the connection being attempted; to remove the CONN: prefix; to increase the number of TCP/IP messages.
- ◆ **Databases can hold more procedures** The primary key values for the SYSPROCEDURE, SYSPROCPARM, SYSPROCPERM, and SYSTRIGGER system tables have been changed from SMALLINT to UNSIGNED INT. This change increases the number of procedures that a database can hold.
 For more information about the number of procedures a database can hold, see “Size and number limitations” [[SQL Anywhere Database Administration Guide, page 504](#)].
To use this feature, you must upgrade the database file format.
- ◆ **Monitoring query performance** New system procedures and utilities have been included to measure query performance.
 For more information, see “sa_get_request_profile system procedure” [[SQL Anywhere Reference, page 862](#)], “sa_get_request_times system procedure” [[SQL Anywhere Reference, page 863](#)], and “Monitoring query performance” [[SQL Anywhere SQL User's Guide, page 254](#)].
- ◆ **New diagnostic properties** Properties allow you to obtain information about connections, databases, and the current database server. The following connection properties have been added in this release:
 - UtilCmdsPermitted property
 - TempTablePages property
 - LastStatement property
 - PacketSize property
 - max_plans_cached property
 - QueryCachePages property

- QueryLowMemoryStrategy property

☞ For more information, see “[Connection-level properties](#)” [*SQL Anywhere Database Administration Guide*, page 458].

The following database properties have been added in this release:

- DBFileFragments property
- LogFileFragments property
- BlobArenas property
- SeparateForeignKeys property
- VariableHashSize property
- TableBitMaps property
- FreePageBitMaps property
- SeparateCheckpointLog property
- Histograms property
- LargeProcedureIDs property
- PreserveSource property
- TransactionsSpanLogs property
- Capabilities property
- TempTablePages property
- CompressedBTrees property
- ProcedurePages property
- QueryCachePages property
- QueryLowMemoryStrategy property

☞ For more information, see “[Database-level properties](#)” [*SQL Anywhere Database Administration Guide*, page 492].



The following server properties have been added in this release:

- MachineName property
- IsJavaAvailable property
- PlatformVer property


☞ For more information, see “[Server-level properties](#)” [*SQL Anywhere Database Administration Guide*, page 481].

- ◆ **Additional performance monitor statistics** Several performance monitor statistics have been added for this release.

☞ For more information, see “[Database performance statistics](#)” [*SQL Anywhere SQL User's Guide*, page 242].

- ◆ **Login procedure allows connections to be disallowed** The `login_procedure` option allows a stored procedure to be called for each new connection. This procedure can now be used to disallow database connections.
 For more information, see “[login_procedure option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 409].
- ◆ **dbsvc enhancements** The `dbsvc` utility for managing Windows services has been extended to list service name used to start and stop the service with the system **net start** and **net stop** commands, and to handle dependencies on other services and groups.
 For more information, see “[The Service utility](#)” [*SQL Anywhere Database Administration Guide*, page 616].
- ◆ **Source format preserved for stored procedures** The source format, including spaces and line breaks, is now stored in the database as a comment. This comment is used for procedure profiling.

Client/server connections

- ◆ **Improved buffer size negotiation** Buffer sizes can now be specified separately for both the client and the server.
To use this feature, you must use version 8 software at both the client and the server. You do not need to upgrade the database.
- ◆ **Communication compression** A new type of communication compression can lead to improved performance if you are transferring data across networks with limited bandwidth, including some wireless networks, some modems, serial links and some WANs.
For more information, see “[Adjusting communication compression settings to improve performance](#)” [*SQL Anywhere Database Administration Guide*, page 112].
To use this feature, you must use version 8 software at both the client and the server. You do not need to upgrade the database.
- ◆ **Enhanced dbping** The `dbping` utility has additional options to help diagnose connection problems. These include the ability to use ODBC to connect, and the ability to report connection, database, and server properties upon connection.
 For more information, see “[The Ping utility](#)” [*SQL Anywhere Database Administration Guide*, page 603].

- ◆ **Suppress TDS debugging option** The `suppress_tds_debugging` option controls whether TDS debugging information appears in the Server Messages window.
 ☞ For more information, see “[suppress_tds_debugging option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 443].
- ◆ **PrefetchBuffer connection parameter** This connection parameter lets you specify the maximum amount of memory for storing prefetched rows.
 ☞ For more information, see “[PrefetchBuffer connection parameter \[PBUF\]](#)” [*SQL Anywhere Database Administration Guide*, page 227].
- ◆ **PrefetchRows connection parameter** The `PrefetchRows` connection parameter lets you specify the maximum number of rows to prefetch when querying the database. In some circumstances, increasing the number of rows prefetched from the database server by the client can improve query performance.
 ☞ For more information, see “[PrefetchRows connection parameter \[PROWS\]](#)” [*SQL Anywhere Database Administration Guide*, page 229].
- ◆ **Client can specify idle timeout** Each client can specify its own idle timeout using the `IDLE` connection parameter. Previously, all connections to a server used the same idle timeout which was specified by the `-ti` server command line option.
 ☞ For more information, see “[Idle connection parameter](#)” [*SQL Anywhere Database Administration Guide*, page 221].

Java in the database

Java in the database includes the following new features:

- ◆ **Java 2 support** Java in the database can now use classes from Java 2 (JDK 1.2 and 1.3) and Java.
 To use this feature, you must upgrade the database using `ALTER DATABASE` or by using the `dbupgrad` utility and supplying the `-jdk` option.
- ◆ **JDBC 2.0** Java classes in the database can now use the JDBC 2.0 interface to access data.
 To use this feature, you must upgrade the database using `ALTER DATABASE` or by using the `dbupgrad` utility and supplying the `-jdk` option.
- ◆ **Diagnostic procedure** A new system procedure, `sa_java_loaded_classes`, lists all classes loaded by the Java virtual machine.

☞ For more information, see “[sa_java_loaded_classes system procedure](#)” [*SQL Anywhere Reference*, page 871].

To use this feature, you must upgrade the database.

- ◆ **Security manager** You can use a built-in security manager or provide your own implementation to control access to security-sensitive Java features.

☞ For more information, see “[Security management for Java](#)” [*SQL Anywhere Programming Guide*, page 100].

Documentation

Several new features have been added to the Adaptive Server Anywhere documentation set to help you find, access and use the information more quickly.

- ◆ **Re-organized books** There have been two major changes to the documentation set since the last release:
 - The *Replication and Synchronization Guide* has been split into two books, describing each of the two synchronization technologies separately. These new books are the *MobiLink Synchronization User's Guide* and the *SQL Remote User's Guide*.
 - The *Adaptive Server Anywhere User's Guide*, *Programming Interfaces Guide*, and *Reference Manual* have been replaced by a *Database Administration Guide*, a *SQL User's Guide*, a *SQL Reference Manual*, and a *Programming Guide*. The database error messages have been moved into their own book. The new organization makes each book a more manageable size in printed form.
- ◆ **New context-sensitive Help** All the user-interface tools, including Sybase Central, Interactive SQL, the Adaptive Server Anywhere debugger, and the Query Editor, share a common cross-platform context-sensitive help system, complete with links to the online books.
- ◆ **Enhanced online books** The HTML Help version of the online books includes a menu bar for quick access to SQL Anywhere Web links, tutorials, procedures, and more.

Miscellaneous

- ◆ **Connections persist across hibernation times** Connections from embedded SQL, ODBC or OLE DB clients now persist while a computer hibernates. Previously, TCP/IP connections between a client and a server on the same computer would be dropped when the computer was woken from hibernation if the computer hibernated for longer than the liveness or idle timeout time.

- ◆ **Viewing current license information** The `dblic` utility now accepts an argument that allows you to view current license information for a server executable without starting the server.

☞ For more information, see “The Server Licensing utility” [[SQL Anywhere Database Administration Guide](#), page 613].

- ◆ **Viewing collation label and name for custom collations** The `dbinfo` utility now returns the collation label and name for custom collations. As well, two new fields, `collationnamebuffer` and `collationnamebufsize`, have been added to the `a_db_info` structure in `dbtools.h`.

- ◆ **sp_remote_tables system procedure** A new argument, **tabletype**, has been added to the `sp_remote_tables` stored procedure. This argument returns the remote table's type.

☞ For more information about the **tabletype** argument, see “`sp_remote_tables` system procedure” [[SQL Anywhere Reference](#), page 932].

- ◆ **-ct command line option** Using the `-ct` command line option, you can turn character set conversion on and off. Character set conversion is now enabled by default, and to turn it off, you can specify `-ct-`. To turn character set conversion on, use `-ct+`.

- ◆ **Obtain remote table foreign key information** Two new stored procedures, `sp_remote_exported_keys` and `sp_remote_imported_keys`, allow you to obtain information about foreign keys and their corresponding primary keys for remote tables.

☞ For more information, see “`sp_remote_exported_keys` system procedure” [[SQL Anywhere Reference](#), page 927] and “`sp_remote_imported_keys` system procedure” [[SQL Anywhere Reference](#), page 929].

- ◆ **xp_sendmail** There are now extended stored procedures for sending email over SMTP as well as MAPI. For more information, see “`xp_startsmtp` system procedure” [[SQL Anywhere Reference](#), page 941] and “`xp_stopsmtp` system procedure” [[SQL Anywhere Reference](#), page 947].

The `xp_sendmail` stored procedure now accepts messages of any length. The length of the long `VARCHAR` parameters for the procedure is limited to the amount of memory available on your system.

☞ For more information, see “`xp_sendmail` system procedure” [[SQL Anywhere Reference](#), page 942].

- ◆ **Replication Server 12 feature for the log transfer manager** The **qualify_table_owner** parameter in the LTM configuration file provides support for the Replication Server 12 feature allowing the table names,

owners, and column names in the primary databases to be different from the replication databases.

☞ For more information, see [“The LTM configuration file” \[SQL Anywhere Database Administration Guide, page 592\]](#).

- ◆ **ASANYSH8 environment variable** A new environment variable, ASANYSH8, has been added. Interactive SQL, Sybase Central, the Adaptive Server Anywhere Console utility, and the debugger use this environment variable to locate the shared components directory.

☞ For more information about the ASANYSH8 environment variable, see [“Setting environment variables” \[SQL Anywhere Database Administration Guide, page 266\]](#).

MobiLink new features

The following is a list of changes and additions to the software introduced in version 8.0.

Flexibility

- ◆ **Java synchronization logic** Synchronization scripts can now be implemented in Java instead of or in addition to the SQL language. These scripts are run in an external JRE using the MobiLink Java environment.

☞ For more information, see [“Writing Synchronization Scripts in Java” \[MobiLink Administration Guide, page 473\]](#).

- ◆ **Synchronization using publications** All the data in a MobiLink client no longer needs to be synchronized at the same time. Rather, data can be organized into publications and each publication synchronized independently. A new syntax for publications and synchronization subscriptions is provided, that is simpler and more precise than the previous syntax.

☞ For more information, see [“SQL Anywhere Clients” \[MobiLink Clients, page 73\]](#).

- ◆ **Configuring Web servers to handle MobiLink synchronization** You can now carry out HTTP synchronization with the MobiLink synchronization server behind a firewall. A Web server plug-in for popular Web servers allows you to carry out HTTP synchronization through Web servers.

☞ For more information, see [“Synchronizing Through a Web Server with the Redirector” \[MobiLink Administration Guide, page 181\]](#).

- ◆ **ActiveSync support for Windows CE clients** Both Adaptive Server Anywhere and UltraLite Windows CE MobiLink clients can use the Windows CE ActiveSync synchronization software.

☞ For more information, see [“Using ActiveSync synchronization”](#) [*MobiLink Clients*, page 99].

- ◆ **Enhanced client command line functionality** You can specify extended options in both `CREATE/ALTER SYNCHRONIZATION SUBSCRIPTION` statements and on the command line.

☞ For more information, see [“dbmlsync syntax”](#) [*MobiLink Clients*, page 109].

- ◆ **Extended options can be stored in the database** Using the `CREATE/ALTER SYNCHRONIZATION SUBSCRIPTION` statements it is possible to store extended options and connection parameters in the database and associate them with subscriptions, users or publications. *Dbmlsync* now reads this information from the database.

☞ For more information see [“dbmlsync syntax”](#) [*MobiLink Clients*, page 109].

Performance

- ◆ **Statement-based uploads** MobiLink now allows statement-based uploads that are not only more intuitive than cursor-based uploads, but also significantly faster. Statement-based uploads employ the **upload_insert**, **upload_delete**, **upload_update**, **upload_new_row_insert**, and **upload_old_row_insert** events. **upload_fetch** script is used for conflict resolution.

☞ For more information see [“Writing scripts to upload rows”](#) [*MobiLink Administration Guide*, page 265].

- ◆ **Multi-processor administration** MobiLink has a new option for setting the maximum number of processors to use. The `-zt` option provides for greater control of the resources used by the MobiLink synchronization server. It can also help to discover and/or work around an ODBC driver with multi-processor issues.

☞ For more information see [“-zt option”](#) [*MobiLink Administration Guide*, page 100].

- ◆ **Optional download acknowledgement** The MobiLink synchronization client can now synchronize without a download acknowledgement, so that the MobiLink synchronization server worker thread does not need to wait for the client to apply the download, freeing up the worker thread sooner for its next synchronization. Download acknowledgement is now an option. Eliminating the download acknowledgement can improve throughput, particularly for slower clients. Note that without a download acknowledgement, the consolidated side will not know that the download succeeded until the next synchronization.

☞ For more information, see [“dbmlsync extended options”](#) [*MobiLink Clients*, page 126].

- ◆ **Buffered download stream** The MobiLink synchronization server now buffers the download stream in a download cache. Since acknowledgement is not required from the client to commit the download transaction, the buffered download stream is sent to the client after the commit. The download transaction is no longer potentially held up by network delays.

The download stream can also be buffered at Adaptive Server Anywhere clients. The size of the buffer available can be set using the *dbmlsync* DownloadBufferSize extended option.

☞ For more information, see “[dbmlsync extended options](#)” [*MobiLink Clients*, page 126].

- ◆ **Bulk loading of connection and table scripts** The first connection or table script requested for a specific table to version_id pairing will cause a bulk load of all the scripts into the cache. The result is improved performance by getting all the scripts in bulk rather than individually.

- ◆ **MobiLink synchronization server shutdown enhancements** You can tell *dbmlstop* to wait until the MobiLink synchronization server is completely shutdown before proceeding. You can also use *dbmlstop* to stop a specific MobiLink synchronization server by name.

☞ For more information, see “[MobiLink stop utility \[mlstop\]](#)” [*MobiLink Administration Guide*, page 584].

- ◆ **Connection timeout** MobiLink database connections that are unused for a specified amount of time are now disconnected automatically by the server. The timeout can be set using the -ct (connection timeout) command line option.

☞ For more information, see “[-ct option](#)” [*MobiLink Administration Guide*, page 48].

- ◆ **Maximum number of concurrent uploaders option** The -wu command line option can set the maximum number of worker threads allowed to upload concurrently, resulting in, for some deployments, increased throughput.

☞ For more information, see “[-wu option](#)” [*MobiLink Administration Guide*, page 87].

Security

- ◆ **MobiLink user authentication** A password-based system for user authentication adds additional security to your MobiLink installation. Now, using -zu, you can allow automatic addition of users when the authenticate_user script is undefined. This allows for user schema information to be used as MobiLink authentication.

☞ For more information, see “[MobiLink Users](#)” [*MobiLink Clients*, page 9].

- ◆ **MobiLink user administration** The dbmluser utility has been extended to allow users to be deleted from the system as well as added. Other refinements have been made to this utility. The dbmluser command line options -pf, -pp, and -pu have been deprecated and replaced with -f, -p, and -u respectively.

☞ For more information, see “[MobiLink user authentication utility \[mluser\]](#)” [*MobiLink Administration Guide*, page 586].

Enhanced reporting

- ◆ **Statistical scripts** MobiLink now has scripts for tracking synchronization statistics. Once gathered, these synchronization statistics may be used for monitoring the performance of your synchronizations.

☞ For more information, see “[synchronization_statistics connection event](#)” [*MobiLink Administration Guide*, page 428], “[synchronization_statistics table event](#)” [*MobiLink Administration Guide*, page 431], “[upload_statistics connection event](#)” [*MobiLink Administration Guide*, page 457], and “[upload_statistics table event](#)” [*MobiLink Administration Guide*, page 462].

- ◆ **Detailed network error information** The MobiLink synchronization server and client now display detailed error information along with error codes to help you better resolve any errors as they arise. You will see the network layer reporting the error, the network operation being performed, the error itself and a system-specific error code.

- ◆ **Remote Adaptive Server Anywhere output log sent to MobiLink synchronization server on error** Troubleshooting synchronization problems is simplest when both the remote log and the MobiLink synchronization server log are available for inspection. This new feature sends the Adaptive Server Anywhere remote’s output log up to the MobiLink synchronization server when a client-side error occurs.

☞ For more information, see “[-e option](#)” [*MobiLink Administration Guide*, page 53].

- ◆ **Log messages identify the worker thread** Messages displayed to the MobiLink synchronization server log now indicate the worker thread that logged the message. This makes it possible to distinguish messages that are due to the same user attempting to synchronize concurrently. It also helps distinguish messages when the same user synchronizes twice without delay.

- ◆ **Verbose logging** You can use additional modifiers on the MobiLink synchronization server -v command line option to configure MobiLink synchronization server logging.

☞ For more information, see “-v option” [*MobiLink Administration Guide*, page 83].

- ◆ **Ignored rows are reported to clients** If the MobiLink synchronization server ignores any uploaded rows because of absent scripts, a messages is returned to the client. The message is displayed as a warning by Adaptive Server Anywhere clients, and in the `ignored_rows` synchronization parameter in UltraLite clients.

☞ For more information, see “Ignored Rows synchronization parameter” [*MobiLink Clients*, page 387].

Ease of use

- ◆ **Last download timestamp** The last download timestamp is written to the MobiLink client database automatically.
- ◆ **Automatic synchronization script generation** MobiLink can be instructed to generate scripts suitable for snapshot synchronization. The `-za` option controls creation and activation of these scripts.
- ◆ **Example synchronization script generation** MobiLink can be instructed to generate example synchronization scripts. The `-ze` command line option is used to control whether example scripts are to be generated.

Adaptability

- ◆ **Support for popular RDBMSs** As consolidated databases, MobiLink now supports Oracle 8i and 9i, Microsoft SQL Server 7, Microsoft SQL Server 2000, IBM’s DB2 and more.
- ☞ For more information, see “ODBC drivers supported by MobiLink” [*MobiLink Administration Guide*, page 648].
- ◆ **Liveness detection in TCP/IP streams** The TCP/IP-based streams that are used during MobiLink synchronization now accept a new parameter, both on the client and server side, called **keep_alive**, that enables liveness checking.

☞ For more information, see “-x option” [*MobiLink Administration Guide*, page 88].

UltraLite new features

UltraLite 8.0 introduces the following new features:

Security

- ◆ **User authentication** In previous releases, UltraLite databases had no user authentication mechanism to govern access. In this release, a built-in user authentication mechanism is provided. Unlike user IDs for most relational database management systems, the UltraLite user IDs do not imply any ownership of tables and other database objects.
- ◆ **Database encryption** You can improve the security of your data by encrypting your database. Two methods are supplied.

Synchronization

- **Strong encryption** The database can be encrypted using a strong encryption algorithm for maximum security. There is a performance penalty to pay for this security. The encryption is key-based and uses the AES 128-bit algorithm.
- **Database obfuscation** You can improve the security of your data by obfuscating the database. Without obfuscation, the data in the database is viewable using a tool such as a hex editor. Obfuscation prevents casual attempts at viewing data but does not offer the watertight protection of strong encryption. Obfuscation does not have the performance penalty that strong encryption carries.
 ☞ For more information, see [“Security considerations” \[UltraLite Database User's Guide, page 46\]](#).
- ◆ **Secure synchronization for UltraLite Java applications** Secure synchronization using Certicom transport-layer security was previously available only from C/C++ UltraLite applications. It is now available from UltraLite Java applications.
- ◆ **ActiveSync synchronization** UltraLite applications on Windows CE devices can use ActiveSync to synchronize.
 ☞ For more information, see [“Adding ActiveSync synchronization to your application” \[UltraLite C/C++ User's Guide, page 112\]](#).
- ◆ **More flexible synchronization** Several new features have been added to enable more efficient and flexible selection of data to synchronize:
 - You can use publications to partition your data into different sets, which can be synchronized separately. This permits the efficient synchronization of time-sensitive data, perhaps over slow connection links, while other data can be synchronized at a more convenient time.
 - Download-only synchronization permits you to add read-only tables to your UltraLite database, and to synchronize them efficiently using a download-only synchronization.
 - You can mark a table to be synchronized each time, whether or not the data in the table has changed. This feature allows you to maintain user-configurable information on the UltraLite client that controls synchronization.
 ☞ For more information, see [“Using the allsync table suffix” \[MobiLink Clients, page 352\]](#).
- ◆ **Global autoincrement default column values** This feature provides a straightforward way of maintaining primary key uniqueness in a synchronizing database.
 ☞ For more information, see [“Overriding partition sizes for autoincremented columns” \[MobiLink Clients, page 360\]](#).

- ◆ **Additional control for UltraLite generator** New command line options have been added for the `ulgen` and `sqlpp` executables:
 - **Script version** You can associate a script version with generated synchronization scripts.
 - **Log query execution plans** The query execution plans for generated queries can be exported and displayed in Interactive SQL.
 - ◆ **Error reporting** The `stream_error` field on the `ul_synch_info` structure can be used to determine the cause of synchronization errors.
 - ☞ For more information, see [“Stream Error synchronization parameter” \[MobiLink Clients, page 396\]](#).
- Database management
- ◆ **Re-use of existing databases** In previous releases of UltraLite, any change to a database application required a rebuild and synchronization of the database. With this release, you can continue to use an UltraLite database with a new version of your application as long as the database schema does not change. Changes to queries do not of themselves require a new database, unless they reference new columns and so change the schema of the generated database.
 - ◆ **Database defragmentation** The UltraLite store is designed to efficiently reuse free space, so that explicit defragmentation is not required under normal circumstances. For applications with extremely strict space requirements, an explicit defragmentation function is provided.
 - ◆ **Choice of page size** You can choose to use 2 KB page sizes as an alternative to the default 4 KB pages.
- Development features
- ◆ **CodeWarrior 7 support** The UltraLite plugin for CodeWarrior now supports CodeWarrior version 7.
 - ◆ **eMbedded Visual C++** Development using this tool is supported, and an eMbedded Visual C++ project is supplied for the CustDB sample application.
 - ◆ **Palm OS 4.0 and file-based data storage** UltraLite now supports version 4.0 of the Palm Computing Platform. Beginning with Palm 4.0, a variety of secondary storage schemes is introduced. You can use a file-based UltraLite data store on an expansion card for a Palm 4.0 device.
 - ☞ For more information, see [“ULEnableFileDB function \(deprecated\)” \[UltraLite C/C++ User’s Guide, page 165\]](#).
 - ◆ **Improved synchronization for Palm Computing Platform** A new and simplified synchronization mechanism for HotSync and ScoutSync synchronization on the Palm Computing Platform has several benefits over previous synchronization mechanisms:

- Launch and exit times are fast.
- No extra storage is required on the Palm device during synchronization.
- The application can be synchronized several times without launching.
- No stream parameter needs to be specified.

The **ULPalmDBStream** and **ULConduitStream** functions are deprecated.

☞ For more information, see [“Adding HotSync synchronization to Palm applications”](#) [*UltraLite C/C++ User's Guide*, page 92].

- ◆ **Easier deployment on the Palm Computing Platform** You can deploy initial copies of the UltraLite database to your end users so that the first synchronization does not have to download an initial copy of the data for each user.

☞ For more information, see [“Deploying Palm applications”](#) [*UltraLite C/C++ User's Guide*, page 95].

- ◆ **Improved handling of Palm segments** When developing for the Palm Computing Platform, application code must be divided into segments of limited size.

The segmentation method provided in earlier versions of the software allowed no user control over the segmentation of the UltraLite generated code, and tended to assign too many segments (which could degrade performance). A new mechanism generates fewer segments and provides customers with control over the assignment of segments.

- ◆ **LONG values in embedded SQL** You can use host variables for long values (between 32 KB and 64 KB) using the `DECL_LONGVARCHAR` and `DECL_LONGBINARY`.
- ◆ **Analyzer hooks in the reference database** The UltraLite generator now invokes stored procedures before and after the analysis process.
- ◆ **Query plan information** The UltraLite generator can now output the access plan to be used for queries in UltraLite applications. Also, you can view the access plan that would be used for UltraLite from Interactive SQL.
- ◆ **Script version control** You can specify the script version to be used for synchronization on the UltraLite generator command line.
- ◆ **Additional SQL and API features** The following features are now available to UltraLite applications.

- **@@identity supported** The @@identity global variable is now supported by UltraLite. This feature is useful in the context of global autoincrement default column values. In the C++ API, use the **ULConnection::GetLastIdentity()** method.
- **Number of rows in a table** From the C++ API programming interface you can determine the number of rows in a table using the **ULTable::GetRowCount()** method. Embedded SQL users continue to use the `SELECT COUNT(*) FROM table-name` statement.
- **Delete all rows in a table** From the C++ API programming interface you can delete all rows in a table using the **ULTable::DeleteAllRows()** method. Embedded SQL users continue to use the `DELETE FROM table-name` statement.
- **Number of rows affected** From embedded SQL you can determine the number of rows affected by the last INSERT, UPDATE, or DELETE statement using the `SQLCOUNT` macro.
- **Number of rows to be uploaded** You can determine the number of rows that need to be synchronized.
☞ For more information, see [“ULCountUploadRows function” \[UltraLite C/C++ User's Guide, page 289\]](#).
- **Last download time** You can obtain the last download time of a publication from the UltraLite application.
☞ For more information, see [“ULGetLastDownloadTime function” \[UltraLite C/C++ User's Guide, page 292\]](#).
- **Additional cursor operations** The **ULTable** class of the C++ API has additional methods (**FindFirst**, **FindNext**, **FindPrevious**, **FindLast**) to locate rows in a result set.
- **Queries from DUMMY system table** Queries of the form `SELECT ... FROM DUMMY` are now supported.
- **Updating multiple tables** Cursors over multiple tables can now accept updates that modify more than one table.
- **Improved LONG data type handling for embedded SQL** The `DECL_LONGVARCHAR` and `DECL_LONGBINARY` host variable types can be used to send or retrieve data over 32 KB in a single operation.
☞ For more information, see [“Data types in Embedded SQL” \[UltraLite C/C++ User's Guide, page 42\]](#).

SQL Remote new features

- ◆ **Event-hook procedures** A set of event-hook procedures have been added to enable customization of the replication process. By writing

stored procedures with specified names, you can add customizations at several points in the actions the Message Agent takes during replication.

☞ For more information, see [“SQL Remote event-hook procedures”](#) [*SQL Remote User's Guide*, page 209].

Behavior changes in version 8

This section lists the behavior changes introduced in components of SQL Anywhere Studio version 8.

Adaptive Server Anywhere behavior changes

The following are behavior changes from previous versions of the software.

☞ For a list of newly deprecated and unsupported features, see [“Deprecated and unsupported features” on page 271](#).

- ◆ **Java in the database separately licensable** As a consequence, the default behavior when creating a database is to exclude support for Java in the database.

Java in the database is no longer needed in UltraLite reference databases, as the UltraLite generator has been changed to use an external Java virtual machine.

☞ For more information, see [“SQL Anywhere components” \[Introducing SQL Anywhere 10, page 8\]](#).

- ◆ **Aggregate functions and outer references** Adaptive Server Anywhere version 8 follows new SQL/99 standards for clarifying the use of aggregate functions when they appear in a subquery. These changes affect the behavior of statements written for previous versions of the software: previously correct queries may now produce error messages, and result sets may change.

☞ For more information, see [“Aggregate functions and outer references” \[SQL Anywhere SQL User’s Guide, page 309\]](#).

- ◆ **User-supplied selectivity estimates** Adaptive Server Anywhere allows you to specify explicit selectivity estimates to guide the choice of access plan. These estimates were most useful as workarounds to performance problems where the software-selected access plan was poor. The new `user_estimates` connection option controls whether the optimizer uses or ignores user-supplied selectivity estimates.

If you have used these estimates as a workaround to performance problems, we recommend setting the `user_estimates` option to `OFF` because an explicit estimate may become inaccurate and may force the optimizer to select poor plans. This version includes query processing enhancements such as internal join algorithms which provide a significant improvement in query performance.

☞ For more information about user-supplied selectivity estimates, see [“user_estimates option \[database\]” \[SQL Anywhere Database Administration](#)

[Guide, page 451](#)] and “Explicit selectivity estimates” [[SQL Anywhere Reference, page 28](#)].

- ◆ **Row ordering** A side-effect of improvements to query processing for version 8.0 is that row ordering is less deterministic. In the absence of an ORDER BY clause, Adaptive Server Anywhere returns rows in whatever order is most efficient. This means the appearance of result sets may vary depending on when you last accessed the row and other factors. The only way to ensure that rows are returned in a particular order is to use ORDER BY.

The LIST function is among those functions particularly affected by this change.

- ◆ **Access plan changes** The access plans selected by this release of Adaptive Server Anywhere are less likely to use indexes than previous releases of the software. Improvements to the efficiency of table scans, together with a more selective cost model used in comparing the cost of access plans, leads to a more accurate assessment of the usefulness of indexes than in previous versions of the software.
- ◆ **Cursor changes** A side effect of cursor enhancements is that the cursors in this version provide behavior closer to defined standards than before. This may produce cursor sensitivity changes for some cursors, as Adaptive Server Anywhere supplies behavior that better matches the expectations of ODBC and other interfaces. For example, embedded SQL SCROLL cursors now disallow prefetching, so that value changes are reflected in the cursor.

This change may affect existing applications that check return codes only for SQL_SUCCESS and not SQL_SUCCESS_WITH_INFO. Applications that check for SQL_SUCCESS_WITH_INFO receive a warning if the cursor behavior is different from that requested. The warning is SQLCODE=121, SQLSTATE 01S02.

Insensitive cursors are not updatable.

☞ For more information, see “[Insensitive cursors](#)” [[SQL Anywhere Programming Guide, page 44](#)].

- ◆ **Stored procedure storage** Stored procedures are now stored as written. Adaptive Server Anywhere does create an internal representation of the procedure, which is used for profiling.
- ◆ **OPEN CURSOR on insert not supported** The ability to open a cursor on an INSERT statement has been dropped. Opening an updatable cursor on a SELECT statement gives the same capabilities in an industry-standard manner.

- ◆ **User-defined functions** User defined function parameters and return values are now cached. If a function is used several times within a SQL statement, the cached parameter values may result in the cached result being used, instead of the function being evaluated again. In previous releases, user-defined functions were re-evaluated each time they were needed. The new behavior provides better performance and more consistent results, but may change results compared to previous releases of the software.

- ◆ **NUMBER(*) function changes** The use of the NUMBER function has been restricted to avoid problematic behavior. NUMBER is intended for use in the select-list of a query, to provide a sequential row-numbering of the result set, and this use is still permitted.

The NUMBER function may now give negative numbers in cases where it previously did not, such as if you carry out an absolute fetch with a value of -1 and then move backward through the cursor. The new behavior corresponds to the ISO/ANSI fetch offset.

Use of the NUMBER function in many circumstances, such as a WHERE clause or a HAVING clause, now gives an error.

☞ For more information, see [“NUMBER function \[Miscellaneous\]” \[SQL Anywhere Reference, page 190\]](#).

- ◆ **Custom collation changes** Previously, the -d option in the Collation utility accepted three parameters; now it accepts only two parameters. The *cust-map-file* parameter is no longer accepted.

As well, the script files *collsqmp.sql* and *custmap.sql* are no longer present and cannot be used for built-in or custom collations, respectively.

For newly-created databases, the SYSCOLLATIONMAPPINGS table contains only one row with the collation mapping. For databases created with previous versions of Adaptive Server Anywhere, this table contains a row for each built-in collation.

- ◆ **Trigger name changes** Trigger names no longer need to be unique across a database. They only need to be unique within the table to which they apply. The syntax of DROP TRIGGER and COMMENT ON TRIGGER has consequently changed so that you can only specify an owner if you also specify a table. This means that older scripts that qualify triggers with only an owner will now result in a “Table not found” error.

- ◆ **Addresses changed in sample database** The addresses in the Adaptive Server Anywhere 9.0 Sample database are different from those in previous releases.

- ◆ **JAR file name for internal JDBC driver changed** The internal JDBC driver classes are now installed as a JAR file named ASAJRT instead of ASAJDBC.
- ◆ **RESTORE DATABASE statement permissions** A connection to the utility database is no longer required to execute a RESTORE DATABASE statement. The permissions required to execute a RESTORE DATABASE statement are controlled by the -gu command line option.

☞ For more information, see “[RESTORE DATABASE statement](#)” [*SQL Anywhere Reference*, page 606].

- ◆ **Return empty string as a NULL string for TDS connections** The `tds_empty_string_is_null` option controls whether the server returns empty strings as a string containing one blank character or a NULL string for TDS connections.

☞ For more information, see “[tds_empty_string_is_null option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 444].

- ◆ **COMMENT statement changed** Previously, the syntax for COMMENT ON INDEX included an optional owner name of the index. The index name can now optionally include the owner and table. The syntax for COMMENT ON INDEX is now

COMMENT ON INDEX [[*owner*.]*table*.]*index-name* **IS** *comment*

☞ For more information, see “[COMMENT statement](#)” [*SQL Anywhere Reference*, page 340].

- ◆ **Character set conversion enabled by default** In previous versions of Adaptive Server Anywhere, character set conversion was turned off by default and you had to specify the -ct command line option to enable character set conversion. Character set conversion is now enabled by default, but can be disabled using the -ct- command line option.

When the server determines that the connection's character set differs from the database's character set, the server applies character set conversion to all the character strings sent to and from the server for that connection.

The server disables character set conversion for a connection when it determines that the database and the connection have equivalent character sets.

In most cases, character set conversion should be enabled. One possible change in behavior occurs when binary data is inserted into a database and is fetched as character data, or vice versa. In this case, the data may not be returned exactly as it was entered because the server applies

character set conversion only to character data. To avoid this problem, applications should not send or fetch character data using a binary type.

- ◆ **CONVERT, timestamp_format and date_format** When using the timestamp_format or date_format options, if you specify a character symbol in mixed case (such as Mmm), Adaptive Server Anywhere now chooses the case that is appropriate for the language that is being used. In addition, the CONVERT function now converts character dates into the case that is appropriate to the language that is being used. For example, in English the appropriate case is May, while in French it is mai.

☞ For more information, see “[date_format option \[compatibility\]](#)” [SQL Anywhere Database Administration Guide, page 388], “[timestamp_format option \[compatibility\]](#)” [SQL Anywhere Database Administration Guide, page 446], and “[CONVERT function \[Data type conversion\]](#)” [SQL Anywhere Reference, page 120].

- ◆ **Change to three-valued Boolean logic** Two-valued Boolean logic applies only to cases of *expr* = NULL, where *expr* refers to a base column or an expression over a base column. Otherwise, three-valued logic applies. The ansinull option now affects only this specific case in the query’s WHERE clause.

- ◆ **Sybase Central and Interactive SQL accept COMMLINKS connection parameter** In previous versions of Adaptive Server Anywhere, Sybase Central and Interactive SQL (the dbisql utility) ignored the COMMLINKS connection parameter. Sybase Central and Interactive SQL now accept this parameter.

As a result of this change, some connection strings may behave differently than in previous versions of Adaptive Server Anywhere. Specifically, if you do not supply COMMLINKS=tcPIP, Interactive SQL and Sybase Central do not look for servers on the network.




☞ For more information, see “[CommLinks connection parameter \[LINKS\]](#)” [SQL Anywhere Database Administration Guide, page 205].

- ◆ **Clients ignore SQLLOCALE environment variable** Clients can use the CharSet connection parameter to specify the character set to be used on a connection. In previous versions of Adaptive Server Anywhere, the CHARSET parameter of the SQLLOCALE environment variable was used to change the client’s default character set if the CharSet connection parameter was not supplied. Clients now ignore the SQLLOCALE environment variable.

- ◆ **Unsupported character sets cause connection failure** Clients can use the CharSet connection parameter to specify the character set to be used on a

connection. However, if the server does not support the requested character set, the connection fails. When a client requested an unsupported character set in previous versions of Adaptive Server Anywhere, the connection succeeded with a warning. If the client does not specify a character set, but the client's local character set is unsupported by the server, the connection succeeds, but with a warning that the character set is not supported.

This behavior occurs in version 8 clients connecting to version 6.x, version 7.x, and version 8 database servers.

- ◆ **Default packet size change** The default packets size for client/server communications has been changed from 1024 bytes to 1460 bytes.
 For more information on packet size, see [“CommBufferSize connection parameter \[CBSIZE\]”](#) [*SQL Anywhere Database Administration Guide*, page 204], and [“-p server option”](#) [*SQL Anywhere Database Administration Guide*, page 162].
- ◆ **dbdsn utility manages Adaptive Server Anywhere data sources only** The dbdsn utility for managing Adaptive Server Anywhere ODBC data sources is now explicitly restricted to Adaptive Server Anywhere data sources only.
- ◆ **login_procedure option requires DBA authority** The login_procedure option can only be set by a user with DBA authority. In previous versions of Adaptive Server Anywhere, DBA authority was not required to set this option. A user with DBA authority can change the setting of this option for other users, but users without DBA authority cannot change their own setting of this option. As a result of this change, the DBA can ensure that a common procedure, if necessary, is executed when a user connects.
 For more information, see [“login_procedure option \[database\]”](#) [*SQL Anywhere Database Administration Guide*, page 409].
- ◆ **ESTIMATE_SOURCE returns new values** The ESTIMATE_SOURCE function returns more detailed values than previously.
 For more information, see [“ESTIMATE_SOURCE function \[Miscellaneous\]”](#) [*SQL Anywhere Reference*, page 147].

Deprecated and
unsupported features

This list includes features that are no longer supported and that impact existing applications.

- ◆ **NetWare 4.10 unsupported** Novell NetWare version 4.11 and later is still supported. Versions 3.x and 4.10 are unsupported.
- ◆ **NetBios unsupported** The NetBios port is no longer supported. If you use NetBios, you should switch to TCP/IP or SPX.

- ◆ **IPX unsupported** The IPX port is no longer supported. If you use IPX, you should switch to SPX or TCP/IP.
- ◆ **Deprecated collations** The following collations are no longer supported. Where indicated, they have been superseded by different collations:

Deprecated	Superseded by
437	437LATIN1
850	850LATIN1
852	852LATIN2
860	860LATIN1
863	863LATIN1
865	865NOR
SJIS	932JPN
SJIS2	932JPN
WIN_LATIN1	1252LATIN1
WIN_LATIN5	1254TRK
Internal	850LATIN1
437EBCDIC	

- ◆ **-e option no longer supported** The -e command line option and the -e option in the Data Source utility, used to encrypt client/server communications, are no longer supported. The -ec option has replaced them. On the server, -ec **simple** uses the same encryption algorithm as -e in previous versions of Adaptive Server Anywhere.
- ◆ **None parameter deprecated** The None parameter for the isql_plan option is no longer supported. The query optimization plan now appears on the Plan tab in the Results pane. When you click the Plan tab, a plan always appears. Previously, the plan appeared in the Messages pane.
- ◆ **WITH HASH SIZE n clause deprecated** The WITH HASH SIZE clause is no longer supported.
- ◆ **max_work_table_hash_size option deprecated** The max_work_table_hash_size option is no longer supported.

- ◆ **max_hash_size option deprecated** The `max_hash_size` option is no longer supported.
- ◆ **SATMP environment variable deprecated** The SATMP environment variable used by UNIX versions of Adaptive Server Anywhere to indicate a directory where temporary files are kept is no longer supported. On UNIX, the ASTMP environment variable can be used to indicate where temporary files are kept.
 ☞ For more information, see “SATMP environment variable” [*SQL Anywhere Database Administration Guide*, page 269].
- ◆ **dbtran -id option removed** The `-id` command line option on the `dbtran` utility is not present in this software.

MobiLink behavior changes

- ◆ **MobiLink Adaptive Server Anywhere client setup** MobiLink clients are now configured using publications and synchronization subscriptions, rather than synchronization definitions.
 ☞ For more information, see “SQL Anywhere Clients” [*MobiLink Clients*, page 73].
- ◆ **Last download timestamp parameter changes scripts** The addition of a new parameter to many scripts makes timestamp-based synchronization easier to implement. The new parameter breaks existing scripts, as it is supplied as the first parameter to many scripts. To continue using existing scripts, change the behavior to supply the last download timestamp as the final parameter by supplying the `-zd` MobiLink synchronization server command line option.
- ◆ **MobiLink shutdown** Previously, `dbmlstop` commands from a remote connection could cause the MobiLink synchronization server to shut down. Now only `dbmlstop` requests from the same machine as the MobiLink synchronization server will cause the MobiLink synchronization server to shut down. The `-zs` option, which would allow `dbmlstop` to stop the server, is no longer required.
- ◆ **Default setting for liveness detection in TCP/IP streams has changed** The default setting for `keep_alive` is now 1 (ON).
- ◆ **MobiLink can hide dbmluser information** The amount of information displayed when the `dbmluser` command line utility is used, such as timestamp, copyright, and other MobiLink synchronization server messages no longer appear by default.

- ◆ **MobiLink user authentication** You must use the `-zu+` option on the MobiLink synchronization server command if you do not use MobiLink user authentication.
- ◆ **Default log extension now .mls** Each file is now named DDMMYYNN.MLS where DD is the day of the month, MM is the month number, and YY is the year in the century. NN is a sequence number that starts at 1 with the first file.
- ◆ **dbmlsync StreamCompression extended option deprecated** This option is now ignored.

UltraLite behavior changes

- ◆ **Required code change for Palm applications** Your code must specify whether to use standard record-based database storage or to use the file-based expansion card storage for Palm Computing Platform version 4.x. You must add a single function call before calling `ULPalmLaunch` (embedded SQL) or `ULData.PalmLaunch` (C++ API). The function calls are as follows:

```
ULEnablePalmRecordDB( &sqlca );
```

or

```
ULEnableFileDB( & sqlca );
```

Supply **ULEnablePalmRecordDB** if you use record-based storage, and **ULEnableFileDB** for file-based storage. If the device does not support file-based storage, `ULPalmLaunch` sets `SQLCODE -82`.

The following environments and/or features are no longer supported by UltraLite:

- ◆ **DOS target platform** DOS is no longer a supported platform.
- ◆ **Metrowerks CodeWarrior 5 development platform** CodeWarrior 6 is now required for UltraLite development.
- ◆ **Palm 2.x no longer supported** UltraLite no longer supports development for Palm OS 2.x devices such as the PalmPilot Professional. Version 3.0 or later is required.
- ◆ **ULPalmDBStream and ULConduitStream deprecated** The new synchronization stream for HotSync or ScoutSync synchronization on the Palm Computing Platform means that the **ULPalmDBStream** and **ULConduitStream** functions are obsolete. They are still accepted, but have no effect.

- ◆ **UltraLite generator uses external Java virtual machine** The UltraLite Analyzer now runs external to the database engine, and so can be used against reference databases even if they are not Java-enabled.
 - ◆ **UltraLite JDBC package name changed** The package name for the UltraLite JDBC functions has been changed from **com.sybase.asa.ultralite.jdbc** to **ianywhere.ultralite.jdbc**. This requires a change to the `import` statements used for UltraLite applications.
 - ◆ **All changes must be committed before download synchronization** Download-only synchronization is no longer an exception to the rule that all changes must be committed before synchronization.
- ☞ You should also check Adaptive Server Anywhere behavior changes, as some may have an impact on your application.

CHAPTER 8

What's New in Version 7.0.3

About this chapter

This chapter provides an overview of the new features and behavior changes introduced in SQL Anywhere Studio version 7.0.3.

Contents

Topic:	page
New features	278
Behavior changes	279

New features

This section introduces the new features in Adaptive Server Anywhere version 7.0.3. It provides a listing of major and minor new features, with cross references to locations where each feature is discussed in detail.

- ◆ **Database properties for blank padding and case sensitivity** You can now use two new properties to determine if your database uses blank padding when comparing strings (BlankPadding) or if your database is case sensitive (CaseSensitive).

☞ For more information see [“Database-level properties” \[SQL Anywhere Database Administration Guide, page 492\]](#).

- ◆ **Server property for C2 security mode** You can now use the new C2 server property to determine whether the database server was started using the -sc option. The -sc option is intended for use in a C2-certified environment.

☞ For more information see [“Server-level properties” \[SQL Anywhere Database Administration Guide, page 481\]](#).

- ◆ **Login procedure allows connections to be blocked** The login_procedure option allows a stored procedure to be called for each new connection. This procedure can now be used to disallow database connections.

☞ For more information see [“login_procedure option \[database\]” \[SQL Anywhere Database Administration Guide, page 409\]](#).

- ◆ **FileDSN now supported on UNIX** The FileDSN connection parameter for ODBC data sources is now supported on UNIX.

Behavior changes

The following is a behavior changes from previous versions of the software.

- ◆ **Load table semantics changed** The LOAD TABLE command now has improved semantics if a column list is specified. A column list must specify each of the columns that exist in the file in the order in which they appear. Column names that do not appear in the list are set to NULL, zero, an empty string, or a default value, depending on the column nullability, data type, and default behavior.

Columns that exist in the input file but which are to be ignored by LOAD TABLE can be specified using the column name **filler()**.

☞ For more information see “[LOAD TABLE statement](#)” [*SQL Anywhere Reference*, page 558].

CHAPTER 9

What's New in Version 7.0.2

About this chapter

This chapter provides an overview of the new features and behavior changes introduced in Adaptive Server Anywhere version 7.0.2.

Contents





Topic:	page
New features in version 7.0.2	282
Behavior changes in version 7.0.2	286

New features in version 7.0.2

This section lists the new features introduced in components of SQL Anywhere Studio version 7.0.2.

Adaptive Server Anywhere new features

This section introduces the new features in Adaptive Server Anywhere version 7.0.2. It provides an exhaustive listing of major and minor new features, with cross references to locations where each feature is discussed in detail.

- ◆ **Dynamic cache sizing** On Windows 95/98, the size of the database server cache increases and decreases depending on the load on the database server and the other demands on system memory. This feature removes the need for choosing an explicit cache size in many circumstances, and can also boost performance.
 For more information, see [“Dynamic cache sizing \(Windows XP/200x\)” \[SQL Anywhere SQL User’s Guide, page 227\]](#).
- ◆ **Viewing current license information** The License [dblic] utility now accepts an argument that allows you to view current license information for a server executable without starting the server.
 For more information, see [“The Server Licensing utility” \[SQL Anywhere Database Administration Guide, page 613\]](#).
- ◆ **Additional collations** There are three new collations available: one to support Russian and Ukrainian (1251CYR, ANSI Code Page 1251), one to support Turkish (1254TRK, ANSI Code Page 1254) and one to support specialty requirements for some German users (1252DEU, ANSI Code Page 1252).
The 1252LATIN1 collation continues to be the recommended German collation. 1252DEU is a specialty collation only, and should not be used without understanding its sorting and comparison properties.
 For a complete list of available collations, see [“Choosing collations” \[SQL Anywhere Database Administration Guide, page 300\]](#).
- ◆ **Interactive SQL return codes** When run from the command prompt, Interactive SQL now sets a program exit code indicating the success or otherwise of the operations in the session.
 For more information, see [“The Interactive SQL utility” \[SQL Anywhere Database Administration Guide, page 582\]](#).

- ◆ **delete_old_logs enhancement** The delete_old_logs database option is used in management of offline transaction logs in a replication environment. The option has been enhanced to permit more control over when processed transaction logs are deleted.
 - ☞ For more information, see “delete_old_logs option [MobiLink] [SQL Remote] [Replication Agent]” [*SQL Anywhere Database Administration Guide*, page 394].
- ◆ **Connection troubleshooting and enhancements** The following changes have been made to permit better troubleshooting and tuning of client/server communications:
 - The APPINFO string is now added to the client debug log file.
 - ☞ For more information, see “AppInfo connection parameter [APP]” [*SQL Anywhere Database Administration Guide*, page 201].
 - Two new connection parameters can be used to tune prefetching of rows.
 - ☞ For more information, see “PrefetchRows connection parameter [PROWS]” [*SQL Anywhere Database Administration Guide*, page 229] and “PrefetchBuffer connection parameter [PBUF]” [*SQL Anywhere Database Administration Guide*, page 227].
 - The **ConnectionName** connection parameter value was previously overridden for ODBC clients. You can now use the **ConnectionName** parameter from ODBC clients.
 - ☞ For a list of connection parameters, see “Connection parameters” [*SQL Anywhere Database Administration Guide*, page 200].
- ◆ **Language Selection utility** The Language Selection (dblang) utility allows you to report and modify the language registry for the Adaptive Server Anywhere messages and Sybase Central interface elements.
 - ☞ For more information, see “The Language Selection utility” [*SQL Anywhere Database Administration Guide*, page 587].
- ◆ **dbspawn enhancement** The Spawn (dbspawn) utility optionally reports the operating system process ID of the database server.
 - ☞ For more information, see “The Start Server in Background utility” [*SQL Anywhere Database Administration Guide*, page 632].
- ◆ **First day of week option** The default first day of week is now 7, which is Sunday. This value affects the result of DATEPART when obtaining a weekday value. You can change the first day of week using the DATEFIRST option in the Transact-SQL SET statement. You can set it permanently using SET OPTION first_day_of_week=*n*.

☞ For more information, see “SET statement [T-SQL]” [*SQL Anywhere Reference*, page 633], or “first_day_of_week option [database]” [*SQL Anywhere Database Administration Guide*, page 397].

- ◆ **New migration tool** You can migrate (import) remote Oracle, DB2, Microsoft SQL Server, Sybase Adaptive Server Enterprise and Sybase Adaptive Server Anywhere databases into Adaptive Server Anywhere using the new sa_migrate set of stored procedures.

☞ For more information, see “Migrating databases to SQL Anywhere” [*SQL Anywhere SQL User’s Guide*, page 695].

- ◆ **Event handlers** Adaptive Server Anywhere can now determine how many instances of a particular event handler is executing at any given time. This is useful for limiting event handlers to only one instance at a time.

☞ For more information, see “EVENT_PARAMETER function [System]” [*SQL Anywhere Reference*, page 150].

- ◆ **New connection property** A new connection property helps distinguish between internal connections used to run event handlers.

☞ For more information, see “CONNECTION_PROPERTY function [System]” [*SQL Anywhere Reference*, page 117].

- ◆ **Dbdsn supports user and system specifiers** The Data Source [dbdsn] utility now supports the u (user) and s (system) options.

☞ For more information, see “The Data Source utility” [*SQL Anywhere Database Administration Guide*, page 552].

- ◆ **Support for comments in @filename files** Adaptive Server Anywhere now supports comment lines in @filename files.


☞ For more information, see “@data server option” [*SQL Anywhere Database Administration Guide*, page 127].

- ◆ **Truncate timestamp option** To allow for greater compatibility with non-Adaptive Server Anywhere databases, you can now truncate timestamp values.

☞ For more information, see “truncate_timestamp_values option [database] [MobiLink]” [*SQL Anywhere Database Administration Guide*, page 447].

- ◆ **Obtaining licensing information** Engine properties have been added to help you obtain accurate licensing information about your copy of Adaptive Server Anywhere.

☞ For more information, see “Server-level properties” [*SQL Anywhere Database Administration Guide*, page 481].

- ◆ **Resetting the autoincrement value** The `sa_reset_identity` system procedure allows you to reset an autoincrement value for the next row.
 For more information, see “[sa_reset_identity system procedure](#)” [*SQL Anywhere Reference*, page 903].

MobiLink new features

- ◆ **Maximum number of threads applying upload streams** To reduce database contention, the `-wu` command line option can now be used to set the maximum number of worker threads allowed to upload concurrently. The upload requests are processed in first-come, first-serve order.
- ◆ For more information, see “[MobiLink Synchronization Server Options](#)” [*MobiLink Administration Guide*, page 31].

Behavior changes in version 7.0.2

This section lists the behavior changes introduced in components of SQL Anywhere Studio version 7.0.2.

Adaptive Server Anywhere behavior changes

The following are behavior changes from previous versions of the software.

- ◆ **Aliases must be defined before first reference** In earlier versions of SQL Anywhere, it was possible to refer to an alias in a SELECT list before the definition of the alias had appeared. An attempt to do so will now generate the error “Definition for alias alias-name must appear before its first reference”. To prevent this error, the SELECT list must be re-ordered so that the alias definition appears before its first use.

CHAPTER 10

What's New in Version 7.0.1

About this chapter

This chapter provides an overview of the new features and behavior changes introduced in Adaptive Server Anywhere version 7.0.1.

Contents

Topic:	page
New features in version 7.0.1	288

New features in version 7.0.1

This section lists the new features introduced in components of SQL Anywhere Studio version 7.0.1.

Adaptive Server Anywhere new features

This section introduces the new features in Adaptive Server Anywhere version 7.0.1. It provides an exhaustive listing of major and minor new features, with cross references to locations where each feature is discussed in detail.

- ◆ **New Service utility** Running a database server as a service under NT allows databases to keep running without tying up the computer on which they are running. Previously, you added services using the Create a New Service wizard from Sybase Central. In Version 7 of Adaptive Server Anywhere, you can now also manage Adaptive Server Anywhere services on Windows NT using the Service Creation [dbsvc] utility. A variety of options allow you to add or delete a service, list all Adaptive Server Anywhere services, or display the details of a particular service. This feature is particularly useful for embedding the creation of a service in installations.

☞ For more information about the Service utility, see “[The Service utility](#)” [*SQL Anywhere Database Administration Guide*, page 616].

- ◆ **Windows CE 3.0 support** In addition to Windows CE 2.11, Adaptive Server Anywhere now supports Windows CE 3.0 on the following processors:

- MIPS
- Hitachi SH3.
- ARM.

Support for Windows CE 2.11 is provided on a wider range of platforms.

With support for Windows CE 3.0, the OLE DB driver on CE works without installing any additional software.

- ◆ **Embedded SQL enhancements** A new function, **db_locate_servers**, provides a programmatic way of locating Adaptive Server Anywhere database servers listening on TCP/IP.

☞ For more information, see “[db_locate_servers function](#)” [*SQL Anywhere Programming Guide*, page 443].

A new callback function, **DB_CALLBACK_CONN_DROPPED**, provides a way of adding logic when the database server is about to drop a connection.

- ☞ For more information, see “[db_register_a_callback function](#)” [*SQL Anywhere Programming Guide*, page 445].
- ◆ **Connection-level Debug and LogFile connection parameters** The DBG and LOG client-side connection parameters are now connection-specific, so you can configure debug information separately for different connections, even from the same application.
 - ☞ For more information, see “[LogFile connection parameter \[LOG\]](#)” [*SQL Anywhere Database Administration Guide*, page 225].
- ◆ **New database property** The **LTMGeneration** property has been added for users of the Replication Agent, or LTM. This property is primarily for use in technical support cases.
 - ☞ For more information, see “[Database-level properties](#)” [*SQL Anywhere Database Administration Guide*, page 492].
- ◆ **New deployment feature** Users of InstallShield Professional 5.5 and up can use the new SQL Anywhere Studio InstallShield Template Projects to deploy their own application. This feature allows you to quickly build your application’s installation using the entire template project, or just the parts that apply to your install.
- ◆ **New backup statement feature** When using the Backup statement, you can specify an empty string as a directory to rename or truncate the log without copying it first. This is particularly useful in a replication environment where space is a concern. You can use this feature with an event handler on transaction log size to rename the log when it reaches a given size, and with the delete_old_logs option to delete the log when it is no longer needed.
 - ☞ For more information, see the “[BACKUP statement](#)” [*SQL Anywhere Reference*, page 318].

MobiLink new features

Following is a list of changes and additions to the software introduced in version 7.0.1.

- ◆ **User authentication** A password-based system for user authentication adds additional security to your MobiLink installation.
 - ☞ For more information, see “[MobiLink Users](#)” [*MobiLink Clients*, page 9].
- ◆ **Extensive documentation of transport-layer security** The transport-layer security documentation has been extended to describe a variety of architectures possible with this powerful security mechanism.

☞ For more information, see “[MobiLink Transport-Layer Security](#)” [[MobiLink Administration Guide](#), page 201].

◆ **Customizing synchronization and synchronization-related processes**

The Adaptive Server Anywhere synchronization client *dbmlsync* now supports a set of events. You can add stored procedures to your Adaptive Server Anywhere database to program event-based actions. This adds flexibility to the synchronization process, including the ability to schedule synchronization.

☞ For more information, see “[Introduction](#)” [[MobiLink Clients](#), page 201].

◆ **Synchronization optimizations** You can optimize the following aspects of the synchronization process.

- UltraLite client applications can specify that a synchronization includes only uploads, and that no download phase should be attempted.

This option lessens the overall synchronization time when only uploads are needed.

- Adaptive Server Anywhere clients can specify an incremental upload option to reduce memory requirements for large uploads.

☞ For more information, see “[Using dbmlsync extended options](#)” [[MobiLink Clients](#), page 93].

- Adaptive Server Anywhere clients can permit concurrent modification of rows during synchronization.

☞ For more information, see “[Concurrency during synchronization](#)” [[MobiLink Clients](#), page 96].

◆ **Scheduling synchronization** You can use an extended option to configure the *dbmlsync* utility or a synchronization definition to synchronize according to a schedule.

☞ For more information, see “[Scheduling synchronization](#)” [[MobiLink Clients](#), page 103].

◆ **Adaptive Server Anywhere client synchronization utility**

enhancements There are several enhancements to the *dbmlsync* utility:

- You can supply the *-mp* and *-mn* options to supply or change the MobiLink password.
- You can supply repeated *-n* options to synchronize more than one synchronization definition.
- The *-v* option now generates more useful information, including options set in the synchronization definition.

- The -r option is extended to allow more flexibility in uploads when the recorded progress indicators in the client and consolidated databases do not match.
- The -x option renames and restarts the transaction log. This option is useful if you use the consolidated database as a backup of the data at the client, so that client-side backups are not required.
- If you do not specify connection parameters on the command line, *dbmlsync* displays a dialog on which you can provide connection parameters and startup options.
- The *dbmlsync* window displays synchronization progress, and allows you to cancel synchronization.

☞ For more information, see “[dbmlsync syntax](#)” [*MobiLink Clients*, page 109].

- ◆ **New MobiLink synchronization server options** The MobiLink synchronization server provides additional options.

☞ For more information, see “[MobiLink Synchronization Server Options](#)” [*MobiLink Administration Guide*, page 31].

- ◆ **New script events** New scripts have been added for handling and reporting errors arising from the ODBC Driver Manager, and to provide additional flexibility when designing synchronization techniques.

☞ For more information, see the following:

- “[handle_odbc_error connection event](#)” [*MobiLink Administration Guide*, page 397]
- “[prepare_for_download connection event](#)” [*MobiLink Administration Guide*, page 417]
- “[report_odbc_error connection event](#)” [*MobiLink Administration Guide*, page 422]

- ◆ **Interface to dbmlsync features** Developers using the C programming language can add features of the *dbmlsync* utility to their application.

☞ For more information, see “[Initiating synchronization from an application](#)” [*MobiLink Clients*, page 97].

SQL Remote new features

SQL Remote version 7.0.1 includes the following new features.

- ◆ **More message links on Novell NetWare** You can now use the FTP and SMTP/POP message links on Novell NetWare.

- ◆ **Enhanced verbose mode** Verbose mode for the Message Agent now writes out full connection information, with user IDs and passwords replaced by asterisks.

UltraLite new features

UltraLite 7.0.1 introduces several new features:

- ◆ **New synchronization stream for Palm Computing Platform** In addition to the current **ULPalmDBStream** synchronization stream, a new synchronization stream is available for the Palm Computing Platform in this release. The new stream is called **ULConduitStream**, and in many circumstances this stream can provide dramatic performance improvements for HotSync synchronization.

This feature superceded

A new conduit-based synchronization stream introduced in version 8.0.0 supercedes both **ULPalmDBStream** and **ULConduitStream**.

- ◆ **Monitoring and canceling synchronization** You can view synchronization status and build the ability to cancel synchronization into your UltraLite applications.
- ◆ **User authentication in MobiLink** MobiLink synchronization now has its own user authentication scheme. Password fields and methods have been added to the UltraLite synchronization parameters to take advantage of this scheme.

☞ For more information, see “[Network protocol options for UltraLite synchronization streams](#)” [*MobiLink Clients*, page 406].

- ◆ **New platforms for secure synchronization** You can now use the transport-layer security features for synchronization from a wider range of target platforms, including Windows CE on the Hitachi SH4 chip, and VxWorks on Intel x86 chips and on the Windows VxSim emulator.

☞ For more information, see the following:

- “[SQL Anywhere supported platforms](#)” [*Introducing SQL Anywhere 10*, page 19]
- “[Network protocol options for UltraLite synchronization streams](#)” [*MobiLink Clients*, page 406].
- “[Synchronization on Windows CE](#)” [*UltraLite C/C++ User’s Guide*, page 112].

VxWorks unsupported in version 9

Support for the VxWorks platform is dropped entirely in version 9.

- ◆ **Non-synchronizing tables** You can include tables in the reference database that are included in the UltraLite database, but are not synchronized. Other than synchronization, the tables can be used like any other table in the remote database.
 - ☞ For more information, see [“Controlling synchronization with allsync and nosync table suffixes in UltraLite”](#) [*MobiLink Clients*, page 351].
- ◆ **Windows CE emulator support enhancements** You can now run UltraLite applications under Windows CE x86 emulators.
- ◆ **Synchronization optimization** Client applications can specify that a synchronization includes only uploads, and that no download phase should be attempted. This option lessens the overall synchronization time when only uploads are needed, especially over slow communication links.
 - ☞ For more information, see [“Network protocol options for UltraLite synchronization streams”](#) [*MobiLink Clients*, page 406].
- ◆ **Automatic HTTP version detection** The MobiLink synchronization server now detects and uses the HTTP version used by each client. This capability renders the **version** parameter on the MobiLink synchronization server -x option redundant.
 - ☞ For information on the MobiLink synchronization server command line, see [“MobiLink Synchronization Server Options”](#) [*MobiLink Administration Guide*, page 31].
- ◆ **Client port specification** You can specify, at a client, a range of ports used by a client during synchronization. This feature can be useful when synchronizing from a client inside a firewall to a MobiLink synchronization server outside.
 - ☞ For more information, see [“Network protocol options for UltraLite synchronization streams”](#) [*MobiLink Clients*, page 406].

CHAPTER 11

What's New in Version 7.0.0

About this chapter

This chapter provides an overview of the new features and behavior changes introduced in Adaptive Server Anywhere version 7.0.0.

Contents

Topic:	page
New features in version 7.0.0	296
Behavior changes in version 7.0.0	307

New features in version 7.0.0

The primary format for the documentation is HTML Help. The HTML Help Home Page gives you easy access to the new features, information about how to contact Sybase, and other starting points for this release.

If you do not have Internet Explorer 4.0 or HTML Help installed on your computer, you will install Windows Help instead of HTML Help. The content is the same except for the HTML Help home page, which is not present in Windows Help.

If you are using Windows Help, you should look at Chapter 1 of *Getting Started with Adaptive Server Anywhere* for information on Adaptive Server Anywhere new features, and at the first chapters of the *UltraLite Developer's Guide* and the *Replication and Synchronization Guide* for information on new features in those technologies.

Adaptive Server Anywhere new features

This section introduces the new features in Adaptive Server Anywhere version 7.0. It provides an exhaustive listing of major and minor new features, with cross references to locations where details of each feature appear in the manuals.

If you have the printed version of this book, and if you do not have the complete SQL Anywhere Studio documentation set, you should look in the online documentation for the detailed description of each feature. To locate the information in the online documentation, go to the index and enter the specified title.

Administration and ease
of use enhancements

- ◆ **Task scheduling and event handling in the database** You can now add scheduled operations to the database. This can be useful for automatic backups, periodic reports to fill summary tables, and other tasks.

The database server can also be instructed to execute event handlers when certain events occur, including disk space thresholds on the drives holding the database file or the transaction log file, or failed connection attempts.

Event handlers can be created and altered using Sybase Central, and can be debugged using the Adaptive Server Anywhere debugger.

☞ For more information see “Automating Tasks Using Schedules and Events” [*SQL Anywhere Database Administration Guide*, page 771], and “CREATE EVENT statement” [*SQL Anywhere Reference*, page 365].

- ◆ **Updated Sybase Central** Sybase Central has been rewritten and contains significant new features. In particular, Sybase Central is now

available from any supported platform, and not just Windows operating systems.

- ◆ **Updated Interactive SQL** The Interactive SQL [dbisql] utility has been enhanced and is now available as a windowed-application from any supported platform.
- ◆ **New validation features** Additional validation of databases is provided by the new `VALIDATE INDEX` statement and by enhancements to the `VALIDATE TABLE` statement. This statement is called both by the Validation [dbvalid] utility, and by the `sa_validate` system procedure. The enhancements are available through all these routes.

☞ For more information, see “[VALIDATE INDEX statement](#)” [*SQL Anywhere Reference*, page 690], and “[VALIDATE statement](#)” [*SQL Anywhere Reference*, page 687].

- ◆ **Lock troubleshooting** A new system procedure, `sa_locks`, provides information on locks in the database. If lock issues are identified, information on the connection processes involved can be found using the **AppInfo** connection property.

☞ For more information, see “[sa_locks system procedure](#)” [*SQL Anywhere Reference*, page 872], and “[AppInfo connection parameter \[APP\]](#)” [*SQL Anywhere Database Administration Guide*, page 201].

- ◆ **Unloading result sets** The new `UNLOAD SQL` statement allows query result sets to be unloaded into a comma-delimited text file.

☞ For more information, see “[UNLOAD statement](#)” [*SQL Anywhere Reference*, page 671].

- ◆ **Validate backup copies of databases** If you backup a database using the `WAIT BEFORE START` clause, the backup copy is created in such a fashion that it can be started in read-only mode and validated.

☞ For more information, see “[BACKUP statement](#)” [*SQL Anywhere Reference*, page 318].

- ◆ **Default global autoincrement** This feature provides an easy way to generate integer keys which are unique across all databases in a SQL Remote replication environment.

- ◆ **Distributed transactions and three-tiered computing** Distributed transactions include operations on more than one server in a single transaction. A transaction server controls the commit and rollback behavior of distributed transactions.

In this release, Adaptive Server Anywhere can participate in distributed transactions coordinated by the Microsoft Distributed Transaction

Integration with
distributed computing
architectures

Coordinator (DTC). Products such as Sybase Enterprise Application Server and Microsoft Transaction Server can use DTC for transaction coordination, so DTC support enables Adaptive Server Anywhere to participate in three-tiered computing with these products.

☞ For more information, see [“Three-Tier Computing and Distributed Transactions”](#) [*SQL Anywhere Programming Guide*, page 59].

Integration with COM

- ◆ **OLE DB provider** OLE DB is a data access model from Microsoft. It uses the Component Object Model (COM) interfaces and, unlike ODBC, OLE DB does not assume that the data source uses a SQL query processor. While it has been possible to access Adaptive Server Anywhere via OLE DB using an OLE DB/ODBC bridge provided by Microsoft, this release of Adaptive Server Anywhere includes an OLE DB provider. This provider brings several benefits:

- OLE DB is the principal data access option for the forthcoming version of Windows CE.
- Some features, such as updating through a cursor, are not available using the OLE DB/ODBC bridge.
- If you use the Adaptive Server Anywhere OLE DB provider, ODBC is not required in your deployment.

☞ For more information, see [“SQL Anywhere OLE DB and ADO APIs”](#) [*SQL Anywhere Programming Guide*, page 285].

Connectivity enhancements

- ◆ **Java connectivity improvements** If you use jConnect to connect to Adaptive Server Anywhere from a Java application, you can now take advantage of many of the features previously available only to ODBC and embedded SQL applications, such as autostarting of database servers, and detailed control over network communications using protocol options.

- ◆ **TCP/IP connectivity** Establishing a client/server connection over TCP/IP is now simpler. Clients no longer need to specify the port number when attempting to connect, even if the server is running on a port other than the default port number (2638). If the default port number is in use when a database server is started, the server acquires an unused port number from the operating system.

If you are trying to connect through a firewall (using UseUDP=NO), and if the database server is not running on port 2638, you must still specify a port number. For more information on this scenario, see [“Connecting across a firewall”](#) [*SQL Anywhere Database Administration Guide*, page 106].

The Server Enumeration utility (dblocate) displays all Adaptive Server Anywhere database servers running TCP/IP on a network. For more information, see [“The Server Enumeration utility”](#) [*SQL Anywhere Database Administration Guide*, page 609].

Performance
enhancements

- ◆ **SPX connectivity** You can use the SPX protocol for connecting to databases. This feature is particularly useful in Novell NetWare environments with IPX/SPX as the primary network protocol. SPX is recommended over IPX.

☞ For more information on SPX at the client, see “[CommLinks connection parameter \[LINKS\]](#)” [*SQL Anywhere Database Administration Guide*, page 205]. For information on SPX on the server, see “[-x server option](#)” [*SQL Anywhere Database Administration Guide*, page 179]. For network protocol options that you can use with SPX, see “[Network protocol options](#)” [*SQL Anywhere Database Administration Guide*, page 233].

- ◆ **Dynamic cache sizing** On Windows NT and UNIX, the size of the database server cache increases and decreases depending on the load on the database server and the other demands on system memory. This feature removes the need for choosing an explicit cache size under in many circumstances, and can also boost performance. On Windows 95/98, a less comprehensive cache resizing is implemented.

☞ For more information, see “[Using the cache to improve performance](#)” [*SQL Anywhere SQL User's Guide*, page 225].

- ◆ **Indexing enhancements** Additional flexibility has been added to control the amount of information stored in indexes (the **hash size**) to improve index selectivity. Also, the architecture of primary and foreign key indexes has been altered.

For indexes on multiple columns, or for indexes on columns in which the first set of characters or digits are similar across many rows, control over hash size provides a way of increasing the selectivity of indexes, and so improving performance.

☞ For more information, see “[Using indexes](#)” [*SQL Anywhere SQL User's Guide*, page 212], “[CREATE INDEX statement](#)” [*SQL Anywhere Reference*, page 382], and “[CREATE TABLE statement](#)” [*SQL Anywhere Reference*, page 427].

☞ For information on how to find the number of levels in an index, see “[sa_index_levels system procedure](#)” [*SQL Anywhere Reference*, page 869].

In previous releases, primary and foreign keys have had a single index automatically associated with them, which describes all primary key values and all the related foreign key entries. In some situations, this architecture lead to poor performance. The new index organization separates these indexes, which leads to improved performance in some situations.

☞ For more information on key indexes, see “[Using keys to improve query performance](#)” [*SQL Anywhere SQL User's Guide*, page 232].

Your database must be unloaded and reloaded to take advantage of variable hash size indexes, and separate key indexes. Running the Upgrade [dbupgrad] utility is not sufficient.

- ◆ **Separate storage for string extensions** The physical storage of values longer than 255 characters has been reorganized. The pages allocated for a table are now divided into two disjoint sets. The first set contain only rows. Where a column value in a row contains a string longer than 255 characters, only a prefix of the string (up to 255 characters) and a reference to a string extension are stored in the row. For strings longer than 255 characters, the string extensions are allocated in the second set of table pages. This change improves performance on queries requiring scans of tables storing long values because a sequential scan of a table only needs to traverse the pages in the first set.

Your database must be unloaded and reloaded to take advantage of this feature.

- ◆ **New database page-sizes** In addition to 1K, 2K, and 4K page sizes, you can now create databases with page sizes of 8K, 16K or 32K. Large page sizes can improve performance in some cases, particularly for large databases. However, there are additional memory requirements with large page sizes, and so they should only be used after investigation of the costs and benefits.

☞ For more information, see “[Initialization utility \(dbinit\)](#)” [*SQL Anywhere Database Administration Guide*, page 573], and “[CREATE DBSPACE statement](#)” [*SQL Anywhere Reference*, page 356].

☞ For information on the number of indexes per table and how it depends on page size, see “[Size and number limitations](#)” [*SQL Anywhere Database Administration Guide*, page 504].

- ◆ **Optimizer tuning** You can use the optimization_goal option to instruct the optimizer to optimize for the time it takes to return the first row of a query, or the overall time it takes to return all rows. The default is to optimize for the first row. If you are using applications such as PowerBuilder DataWindow applications, which require a complete result set, you may want to change this option setting.

☞ For more information, see “[optimization_goal option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 425].

- ◆ **Optimizer enhancements** Further enhancements to the optimizer have been implemented to assist with performance of queries that use internal temporary tables and that use primary and foreign key indexes. These enhancements require no user action.

- ◆ **Larger numbers of users and other identifiers** Many identifiers in the system tables identifying database objects have been changed from SMALLINT to UNSIGNED INTEGER. This change increases the number of objects that can be held in a database without violating an absolute limit.
- ◆ **Inserting and exporting images and documents** Two new system external functions allow you to read and write the contents of files. These functions allow direct inserting of images, documents, and so on into tables from environments such as Interactive SQL.
 - ☞ For more information, see “[Inserting documents and images](#)” [*SQL Anywhere SQL User's Guide*, page 482], “[xp_read_file system procedure](#)” [*SQL Anywhere Reference*, page 936], and “[xp_write_file system procedure](#)” [*SQL Anywhere Reference*, page 938].
- ◆ **New interface for external functions** Stored procedures and user-defined functions that reference external libraries now use a new interface. The new interface provides a wider range of operating systems (including UNIX), a wider range of data types, removes the restriction that returned data fit into 255 bytes, and supports NULL as a valid value for arguments. The older interface is still supported, but should not be used for new development work.
 - ☞ For more information, see “[Creating procedures and functions with external calls](#)” [*SQL Anywhere SQL User's Guide*, page 812].
- ◆ **START DATABASE, STOP DATABASE and STOP ENGINE statements** These statements were previously available only from Interactive SQL. They are now available from all applications.
 - ☞ For more information, see “[START DATABASE statement](#)” [*SQL Anywhere Reference*, page 650], “[STOP DATABASE statement](#)” [*SQL Anywhere Reference*, page 659], and “[STOP ENGINE statement](#)” [*SQL Anywhere Reference*, page 660].
- ◆ **FIRST and TOP clause in updates and deletes** The FIRST and TOP clauses can be used to update or delete only the first one or more of any set of rows satisfying a WHERE clause.
 - ☞ For more information, see “[DELETE statement](#)” [*SQL Anywhere Reference*, page 461], and “[UPDATE statement](#)” [*SQL Anywhere Reference*, page 676].
- ◆ **Explicit table locking** The LOCK TABLE statement allows direct control over concurrency at a table level, independent of the current isolation level.

☞ For more information, see “[LOCK TABLE statement](#)” [*SQL Anywhere Reference*, page 567].

- ◆ **Expressions in Transact-SQL outer joins** The *= and =* operators in a WHERE clause provide a way of specifying outer joins for users who want to use the Transact-SQL dialect. In previous releases, only column names could be used in such joins. Now as long as each side of the join operator refers to a single table, any expression can be used in these joins. For example, the following query is now possible:

```
select *
from customer, sales_order
where substr( customer.id, 1, 1 ) *=
      substr( sales_order.cust_id, 1, 1 )
```

- ◆ **Cursors in stored procedures can reference variables** In stored procedures and user-defined functions, you can declare a cursor on a variable using the following syntax:

DECLARE *cursor-name* **CURSOR USING** *variable-name*

where *variable-name* is a string variable containing the SELECT statement for the cursor.

☞ For more information, see “[DECLARE CURSOR statement \[ESQL \[SP\]\]](#)” [*SQL Anywhere Reference*, page 454].

- ◆ **Additional database and server properties** The following properties have been added:
 - **PageSize** The database server uses a single page size from startup until it is closed down. This page size is the maximum page size database that can be mounted by the database server. You can now obtain this page size using the PageSize server-level property function:

```
select property( 'PageSize' )
```

- **AppInfo** This function provides identification information for a client application. It is a connection property:

```
select connection_property( 'AppInfo' )
```

☞ For more information, see “[AppInfo connection parameter \[APP\]](#)” [*SQL Anywhere Database Administration Guide*, page 201].

- **IsRuntimeServer** This function returns YES if the database server is a limited desktop runtime personal database server. Otherwise, it returns NO.
 - **Log truncation points** Properties for replication-specific log offsets have been added. The properties **LTMTrunc**, **RemoteTrunc**, and **SyncTrunc** return the minimal confirmed log offset for the Replication

Agent, SQL Remote, and MobiLink *dbmlsync* replication, respectively. These offsets are also known as truncation points because they indicate the point at which the transaction log can be truncated. The property **CurrentRedoPos** returns the current offset in the log file, where the next database operation is to be logged.

☞ For a complete list of property functions and information on how to access them, see “[Understanding database properties](#)” [*SQL Anywhere Database Administration Guide*, page 458].

- ◆ **Referential integrity checks before commit** A new system procedure (*sa_check_commit*) allows you to check for referential integrity conflicts before committing changes to a database.

☞ For more information, see “[sa_check_commit system procedure](#)” [*SQL Anywhere Reference*, page 836].

- ◆ **SQL function enhancements** The following functions have been added or enhanced.

- **REPLACE function** This new function replaces all occurrences of a substring with another substring.

☞ For more information, see “[REPLACE function \[String\]](#)” [*SQL Anywhere Reference*, page 212].

- **LIST function enhancement** The LIST function now accepts an optional second value, which is the delimiter string that separates the list items.

☞ For more information, see “[LIST function \[Aggregate\]](#)” [*SQL Anywhere Reference*, page 174].

- ◆ **Output redirection change** The output redirection functionality in Interactive SQL has been extended to include three new Interactive SQL statements and an Export option in the File menu.

You can now use an OUTPUT TO statement to redirect content from the Results pane to a new file. You can add an APPEND clause to append the content to the end of an existing file, or you can add a VERBOSE clause to include the content of the Messages pane with the output.

In earlier versions, output redirection in Interactive SQL could only be done with the symbols >#, >>#, >&, and >>&. You can still use these symbols, but the new Interactive SQL statements allow for more precise output and code that is easier to read.

☞ For more information, see “[Exporting query results](#)” [*SQL Anywhere SQL User's Guide*, page 678].

- ◆ **Embedded SQL enhancements** A new function, **db_string_ping_server**, has been introduced to test that a database server can be located with a specified current connection string.

☞ For more information, see “[db_string_ping_server function](#)” [*SQL Anywhere Programming Guide*, page 451].

- ◆ **New LOAD TABLE / UNLOAD TABLE format** A new format has been added to the UNLOAD TABLE statement to allow data to be output in BCP format and to the LOAD TABLE statement to allow the import of Adaptive Server Enterprise generated BCP out files containing BLOBs.

☞ For more information, see “[LOAD TABLE statement](#)” [*SQL Anywhere Reference*, page 558] or “[UNLOAD TABLE statement](#)” [*SQL Anywhere Reference*, page 673].

- ◆ **Last default timestamp** The new global variable @@dbts returns a TIMESTAMP value that represents the last value generated for a column using DEFAULT TIMESTAMP.

☞ For more information, see “[Global variables](#)” [*SQL Anywhere Reference*, page 38].

- ◆ **Troubleshooting enhancements** On starting the database server, you can log operations executed by the server to a file using the -zr option. You can use the sa_server_option procedure to control the same behavior while the server is running.

☞ For more information, see “[sa_server_option system procedure](#)” [*SQL Anywhere Reference*, page 907], and “[-zr server option](#)” [*SQL Anywhere Database Administration Guide*, page 186].

- ◆ **Archive backup on NetWare** The archive backup format is now supported on NetWare. Archive backups to tape require NetWare 5.

☞ For more information, see “[BACKUP statement](#)” [*SQL Anywhere Reference*, page 318].

- ◆ **Added filtering for dbtran** The command version of the Log Translation [dbtran] utility allows further filtering of the output.

☞ For more information, see “[Log Translation utility \(dbtran\)](#)” [*SQL Anywhere Database Administration Guide*, page 596].



- ◆ **Faster table truncation** The TRUNCATE TABLE statement is much faster for version 7.0 databases, for tables with foreign keys.

- ◆ **Suppressing event log messages** If you run the database server as a Windows NT service, you can suppress event log messages using a registry entry.

☞ For more information, see “[Suppressing Windows event log messages](#)” [*SQL Anywhere Database Administration Guide*, page 126].


SQL Remote new features

SQL Remote version 7.0 includes the following new features.

- ◆ **Globally unique primary keys** You can now use a DEFAULT GLOBAL AUTOINCREMENT column in an Adaptive Server Anywhere database, together with a `global_database_id` option setting in each database, to guarantee unique primary keys throughout a SQL Remote installation of Adaptive Server Anywhere databases. This is a more convenient method than the more manual primary key pool technique.
 For more information, see [“Using global autoincrement default column values”](#) [*SQL Remote User's Guide*, page 93].
- ◆ **Internal unload for dbxtract** The Extract [dbxtract] utility now uses the UNLOAD statement introduced in Adaptive Server Anywhere version 7.0 by default, rather than the slower OUTPUT statement. Options have been introduced to allow you to choose a combination of internal (server-side) and external (client-side) unload and load operations.
 For a complete listing of options, see [“Extraction utility”](#) [*SQL Remote User's Guide*, page 198].

MobiLink and UltraLite new features

Following is a list of changes and additions to the software since version 6.0.3.

- ◆ **Adaptive Server Anywhere clients** MobiLink technology now supports Adaptive Server Anywhere as a client, as well as UltraLite applications.
 For more information, see [“SQL Anywhere Clients”](#) [*MobiLink Clients*, page 73].
- ◆ **mlxtract creates Adaptive Server Anywhere client databases** *mlxtract* creates Adaptive Server Anywhere databases, suitable for use as MobiLink clients, using an Adaptive Server Anywhere reference database as a template.
- ◆ **Synchronization script versions** Synchronization scripts can now be grouped by assigning a script version name with each script. This feature allows the MobiLink synchronization server to respond differently when synchronizing different types of applications, or different versions of the same application.
- ◆ **New data types** LONG BINARY and LONG VARCHAR data types can now be replicated using MobiLink technology.

- ◆ **New HotSync conduit** A new HotSync conduit allows HotSync synchronization with a centrally located MobiLink synchronization server. The MobiLink synchronization server no longer needs to be on the same computer as the HotSync manager is.
- ◆ **ScoutSync conduit** UltraLite applications for the Palm Computing Platform can now synchronize using ScoutSync technology, available from Riverbed Technologies.
- ◆ **report_error script** A new script provides a convenient way to report errors during synchronization. The **report_error** script also makes debugging the behavior of the **handle_error** script much easier. The **report_error** script has the same parameters as the **handle_error** script, except that the first parameter is the action code returned by **handle_error**.

Behavior changes in version 7.0.0

This section lists deprecated and unsupported features, and behavior changes from previous versions of the software.


Adaptive Server Anywhere behavior changes

Deprecated and unsupported features

This list includes features that are no longer supported and that impact existing applications.

- ◆ **Windows 3.x and Windows CE 2.0 no longer supported**
Windows 3.1 and Windows 3.11 are no longer supported. Windows CE 2.0 is no longer supported.
- ◆ **DDE protocol no longer supported** The DDE protocol was used to communicate from 16-bit Windows 3.x applications to a Windows 95/98 database server on the same computer. It is no longer required: Windows 3.x applications based on older versions of the software can use TCP/IP to communicate with the version 7.0 database server.
- ◆ **IPX protocol deprecated** Although communications using IPX are still supported in the present release, it is highly recommended that you use the SPX protocol instead. The protocol options are the same as for IPX, and performance is better. Support for IPX will be dropped in a future release.

By default, both the database server and the client software do not start the IPX protocol unless you instruct it to do so explicitly using the `-x` option or the **CommLinks** connection parameter. The SPX protocol is started by default.

 For information on using SPX from the client side, see “[CommLinks connection parameter \[LINKS\]](#)” [*SQL Anywhere Database Administration Guide*, page 205]. For information on using SPX from the server side, see “[-x server option](#)” [*SQL Anywhere Database Administration Guide*, page 179].
- ◆ **Deprecated network protocol options** The Broadcast and CommAutoStop protocol options are still allowed, but have no effect. They will not be supported in future versions of Adaptive Server Anywhere.
- ◆ **No dbclient compatibility executable** In version 6, the `dbcli6.exe` utility provided easier compatibility with version 5 client connection methods. There is no comparable utility in version 7.

Behavior changes

This list includes behavior changes in existing features that may impact applications or have an impact during development or database management.

- ◆ **Interactive SQL changes** The new version of Interactive SQL has some changes from previous versions. As it is an interactive tool, most do not need documentation.

☞ The supported formats for INPUT and OUTPUT statements have changed, and now include:

- **INPUT** ASCII, DBASE, DBASEII, DBASEIII, EXCEL, FIXED, FOXPRO, LOTUS
- **OUTPUT** ASCII, DBASE, DBASEII, DBASEIII, EXCEL, FIXED, FOXPRO, HTML, LOTUS, SQL

- ◆ **Server name space change** It is now disallowed for more than one database server with the same name to be running on TCP/IP anywhere on the network. Previously, multiple servers with the same name were allowed as long as they were on separate ports.

- ◆ **Mirrored logs deleted when delete_old_logs is On** Previously, any mirror of an old transaction log was not deleted, although the primary copy of the old transaction log was deleted.

- ◆ **ODBC SQLDescribeCol behavior** A `SQLDescribeCol` call on the `@@identity` field now returns `SQL_BIGINT`. In earlier versions, it returned `SQL_INTEGER`.

- ◆ **Update constraints** A new `ansi_update_constraints` option has been added. Setting this option to `Cursors` or `Strict` restricts updates to those allowed by the ANSI standard. Setting this option to `Off`, which is the historical behavior, allows a greater range of updates.




☞ For more information, see [“ansi_update_constraints option \[compatibility\]”](#) [*SQL Anywhere Database Administration Guide*, page 375], and [“UPDATE statement”](#) [*SQL Anywhere Reference*, page 676].

- ◆ **Identifier length limit** Long identifiers are treated more consistently than in the past. Identifiers longer than 128 bytes were sometimes accepted and sometimes not, depending on the type of database object being named. Now any attempt to define identifiers longer than 128 bytes reports an error.

☞ For more information, see [“Identifiers”](#) [*SQL Anywhere Reference*, page 8].


- ◆ **jConnect connections** If you use the `REMOTEPWD` field to connect via jConnect to a named database on an Adaptive Server Anywhere database server, you must assign the field in a different manner for jConnect version 4.2 and above, included with this software.

☞ For more information, see [“Supplying a URL to the driver”](#) [*SQL Anywhere Programming Guide*, page 341].

- ◆ **User-defined errors** Within procedures and triggers, you can declare exceptions in the range 99000 to 99999 as user-defined errors in compound statements. You can use the SIGNAL statement to handle these errors.
 For more information, see “BEGIN statement” [SQL Anywhere Reference, page 324], and “SIGNAL statement” [SQL Anywhere Reference, page 649].
- ◆ **LOAD TABLE and UNLOAD TABLE security** A database server option has been added to control the permissions required to execute the LOAD TABLE and UNLOAD TABLE statements.
 For more information, see “-gl server option” [SQL Anywhere Database Administration Guide, page 150].
- ◆ **@@identity in triggers** If a table (T1) with an autoincrement column has an insert trigger which causes an insert into a second table (T2) also having an autoincrement column, it was not previously possible to obtain the autoincrement value assigned for T1 after the insert had completed. At that point, the value of @@identity would be the value assigned to T2. The behavior of @@identity has been altered to make the value accessible.
 For the new behavior of @@identity within triggers, see “@@identity global variable” [SQL Anywhere Reference, page 43].
- ◆ **Embedded SQL DECL_FIXCHAR** In previous releases, the SQL preprocessor converted a type DECL_FIXCHAR to an array. For example, DECL_FIXCHAR(12) was converted to char name_fixchar[12].
 In the current release, the SQL preprocessor converts DECL_FIXCHAR declarations to DECL_FIXCHAR(12) name_fixchar, and references are of the form name_fixchar.array[i].

SQL Remote behavior changes

The following behavior has changed in version 7.0:

- ◆ **dbxtract uses internal unload** The default behavior of dbxtract is now to use the UNLOAD statement to unload data on the server side, rather than the OUTPUT statement to unload data on the client side. The -ii, -ix, -xi, -xx options allow you to choose which combination of internal and external operations to use, and replace the options -i and -x available in previous releases.
 For a complete listing of options, see “Extraction utility” [SQL Remote User's Guide, page 198].

MobiLink and UltraLite behavior changes

- ◆ **New table and script names** The tables that hold synchronization scripts and related information in the consolidated database now have new names. Previously, these table names began with the prefix **ul_**. This prefix has been changed to **ml_**. Older consolidated databases must be upgraded for compatibility with version 7.0.

Similarly, the stored procedure that facilitates adding table scripts has been renamed from **sp_table_script** to **ml_add_table_script** and the stored procedure that facilitates adding connection scripts has been renamed from **sp_connection_script** to **ml_add_connection_script**.

Under DB2, these names are truncated to 18 characters.

- ◆ **Synchronization scripts require a version name** Synchronization scripts must now be assigned a script version name. Script version names allow the MobiLink synchronization server to treat different clients differently.

CHAPTER 12

What's New in Version 6.0.3

About this chapter

This chapter provides an overview to the new features and behavior changes introduced in Adaptive Server Anywhere 6.0.3.

Contents

Topic:	page
New features in version 6.0.3	312
Behavior changes in version 6.0.3	319

New features in version 6.0.3

This section lists the new features introduced in components of SQL Anywhere Studio version 6.0.3.

Adaptive Server Anywhere new features

In addition to bug fixes, Adaptive Server Anywhere version 6.0.3 includes new features in both the software and the documentation.

- ◆ **Combined stored procedure and Java debugger** The Java debugger that was provided in previous releases has been upgraded. The new version of the debugger is able to debug not only Java classes within the database, but also SQL stored procedures and triggers.

☞ For information on how to use the debugger, see [“Debugging Logic in the Database”](#) [*SQL Anywhere SQL User’s Guide*, page 821].

- ◆ **Read-only databases** You can designate a database as read only when you start a database server. This feature makes deployment of databases on read-only media, such as CD-ROMs, more straightforward.

The **ReadOnly** database property returns On for read-only databases, and OFF for databases that are not being run in read-only mode.

☞ For more information on read-only databases, see [“-r server option”](#) [*SQL Anywhere Database Administration Guide*, page 166].

- ◆ **Computed column extensions** New flexibility has been added to computed columns. You can now add computed columns to non-empty tables, and change the expression associated with a computed column. Computed columns are recalculated in a number of circumstances to ensure that the values are reliable.

☞ For more information, see [“Working with computed columns”](#) [*SQL Anywhere SQL User’s Guide*, page 54], and [“Inserting and updating computed columns”](#) [*SQL Anywhere SQL User’s Guide*, page 56].

☞ For information on syntax, see [“ALTER TABLE statement”](#) [*SQL Anywhere Reference*, page 303].

- ◆ **Support for the euro** Collations have been added that include the euro currency symbol. These collations are the 1252LATIN1 and ISO9LATIN1 collations.
- ◆ **Additional collations** Other collations have been added to the list of supplied collations, including 852POL (OEM Code Page 852 (Latin 2), with Polish ordering), 1250POL (Windows Latin2 code page 1250 with Polish ordering), 1250Latin2 (Windows Latin2 Code page 1250).

932JPN (Japanese), 936ZHO (similar to EUC_CHINA), and 950TAI (similar to EUC_TAIWAN).

☞ For a complete list, see “Choosing collations” [*SQL Anywhere Database Administration Guide*, page 300].

- ◆ **New Windows CE platforms** The SH4 and ARM processors are now supported under Windows CE 2.1x.
- ◆ **ALTER TABLE extensions** The ALTER TABLE statement has been extended to provide SQL/92-compliant clauses to set and drop defaults on columns. These clauses are an alternative to the existing MODIFY clause.

```
ALTER column-name SET DEFAULT default-value
| ALTER column-name DROP DEFAULT
```

☞ For more information, see “ALTER TABLE statement” [*SQL Anywhere Reference*, page 303].

- ◆ **LOAD TABLE extensions** You can now load specific columns of a table using the LOAD TABLE statement. A new CHECK CONSTRAINTS option has been introduced to address rebuild issues.







☞ For more information, see “LOAD TABLE statement” [*SQL Anywhere Reference*, page 558].
- ◆ **Easier connections across firewalls** A set of protocol options has been introduced to allow easier connections across firewalls.

☞ For more information, see “Connecting across a firewall” [*SQL Anywhere Database Administration Guide*, page 106].
- ◆ **BACKUP statement extended** The MATCH keyword has been introduced to allow renaming of the backup copy of the transaction log to a file name of the form *YYMMDDnn.log*. If you use this keyword, you can execute the same statement multiple times without writing over data.

☞ For more information, see “BACKUP statement” [*SQL Anywhere Reference*, page 318].
- ◆ **Easier unload and reload** The Unload [dbunload] utility has been enhanced (-ar option) to allow a single-step unload and reload of a database that can be used whether or not your database is involved in replication.

☞ For more information, see “Unload utility options” [*SQL Anywhere Database Administration Guide*, page 648], and “Rebuilding databases” [*SQL Anywhere SQL User's Guide*, page 685].

- ◆ **Temporary file location** The database server checks for a new environment variable, `ASTMP`, when deciding on the location of the temporary file. This allows you to use directories other than system temporary directories for the temporary file.
 - ☞ For more information, see [“SATMP environment variable” \[SQL Anywhere Database Administration Guide, page 269\]](#).
- ◆ **New system procedures** New system procedures allow DBA users to override some database server options (`sa_server_option`), and to flush the database server cache (`sa_flush_cache`).
 - ☞ For more information, see [“System procedures” \[SQL Anywhere Reference, page 831\]](#).
- ◆ **Character set conversion tuning** You can control the application locale used in character set conversion for an individual connection using the new `CharSet` connection parameter.
 - ☞ For more information, see [“CharSet connection parameter \[CS\]” \[SQL Anywhere Database Administration Guide, page 204\]](#).
- ◆ **Re-organized Performance Monitor statistics** The statistics made available to the Windows NT Performance Monitor have been organized into areas. Some statistics have been added, and ones of little use have been removed.
 - ☞ For a list of available statistics, see [“Monitoring database statistics from Windows Performance Monitor” \[SQL Anywhere SQL User’s Guide, page 240\]](#).
- ◆ **Database properties from the utility database** You can now execute `SELECT` statements, with no tables, against the utility database. This is primarily of use for retrieving database and connection properties.
 - ☞ For more information, see [“Using the utility database” \[SQL Anywhere Database Administration Guide, page 260\]](#).
- ◆ **New database properties** The following properties are available using the `property` function.
 - **IsNetworkServer** Returns YES if connected to a network database server, and NO if connected to a personal database server.
 - ☞ For more information, see [“Server-level properties” \[SQL Anywhere Database Administration Guide, page 481\]](#).
 - **DefaultCollation** You can use the new **DefaultCollation** property to find the default collation to be used when creating a database.
 - ☞ For more information, see [“Finding the default collation” \[SQL Anywhere Database Administration Guide, page 307\]](#), and [“Server-level properties” \[SQL Anywhere Database Administration Guide, page 481\]](#).

- **MultiByteCharSet** You can use the **MultiByteCharSet** database property to determine whether a database is using a multi-byte or single-byte collation.
 For information on this property, see “Database-level properties” [*SQL Anywhere Database Administration Guide*, page 492].
- ◆ **Support for some JDBC 2.0 functions in internal JDBC** The internal server-side JDBC driver now supports functions from the JDBC 2.0 interface. Server-side Java applications can now use features such as scrollable, updatable result sets and batch updates. A side effect is that you can now access result sets from Java methods from Interactive SQL.
- ◆ **Using the main method in Java classes** You can now execute a main method of a Java class from SQL.
 For more information, see “Calling the main method” [*SQL Anywhere Programming Guide*, page 97].
- ◆ **User-defined functions using Java classes** You can wrap a Java method in a SQL user-defined function.
 For more information, see “CREATE FUNCTION statement” [*SQL Anywhere Reference*, page 376].
- ◆ **Extensions to stored procedures using Java methods** You can use OUT and INOUT parameters in stored procedures that are wrappers for Java methods.
 For more information, see “Returning values from Java via stored procedures” [*SQL Anywhere Programming Guide*, page 99].
- ◆ **Multi-threaded Java classes in the database** Support has been added for the package `java.lang.thread`.
 For more information, see “Using threads in Java applications” [*SQL Anywhere Programming Guide*, page 97].
- ◆ **File access from Java** Support has been added for all the classes in the package `java.io`, including those that enable file access from classes in the database. For security reasons, the `java_input_output` option has been introduced, which must be set by the DBA to enable this feature.
 This feature is supported on Windows NT and UNIX.
- ◆ **CONVERT function extensions** The date and time styles supported by the CONVERT function have been extended.
 For more information, see “CONVERT function [Data type conversion]” [*SQL Anywhere Reference*, page 120].

- ◆ **Database server startup dialog** On 32-bit Windows operating systems, if you start a database server with no arguments, a window appears where you can specify a database file and additional parameters.
☞ For more information, see [“Starting the server” \[SQL Anywhere Database Administration Guide, page 8\]](#).
- ◆ **Log Translation [dbtran] utility enhancements** The Log Translation [dbtran] utility permits filtering of the transaction log operations to isolate subsets of operations.
☞ For more information, see [“Log Translation utility \(dbtran\)” \[SQL Anywhere Database Administration Guide, page 596\]](#).
- ◆ **Transaction Log [dblog] utility enhancements** The Transaction Log [dblog] utility now displays additional summary information, including offset information.
☞ For more information, see [“Transaction Log utility \(dblog\)” \[SQL Anywhere Database Administration Guide, page 637\]](#).
- ◆ **Start Server in Background utility enhancements** The Start Server in Background utility (dbspawn) has a -f option to force a server to start even if one is already running. This option uses a ForceStart connection parameter, used only by the db_start_engine Embedded SQL function.
☞ For more information, see [“The Start Server in Background utility” \[SQL Anywhere Database Administration Guide, page 632\]](#), and [“db_start_engine function” \[SQL Anywhere Programming Guide, page 448\]](#).
- ◆ **Replication Agent runs as a daemon** On UNIX operating systems, you can run the Replication Agent as a daemon by supplying the -ud option.
☞ For more information, see [“The Log Transfer Manager” \[SQL Anywhere Database Administration Guide, page 589\]](#).

SQL Remote new features

In addition to bug fixes, SQL Remote version 6.0.3 includes the following new features. Some features in Adaptive Server Anywhere that are particularly relevant to SQL Remote are also included in this list:

- ◆ **FTP and SMTP/POP support on UNIX** The range of message systems supported on UNIX operating systems has been expanded to include FTP and SMTP/POP.
☞ For a listing of supported operating and message systems, see [“Supported Platforms and Message Links” \[SQL Remote User’s Guide, page 221\]](#).

- ◆ **Message link options stored in the database** The message link parameters that control SQL Remote behavior over each message system can now be stored in the database as opposed to the registry. This simplifies deployment and management issues related to message link parameters.
- ◆ **Date and time replication formats** You can now specify database options that instruct SQL Remote what format to use when replicating dates and times. These options are `sr_time_format`, `sr_date_format`, and `sr_timestamp_format`.
 ☞ For more information, see “[Replication of dates and times](#)” [*SQL Remote User's Guide*, page 50], and “[SQL Remote options](#)” [*SQL Remote User's Guide*, page 205].
- ◆ **Message Agent and SQL Remote Open Server run as a daemon** On UNIX operating systems you can run these applications as a daemon using the `-ud` option.
- ◆ **Easier unload and reload of Adaptive Server Anywhere databases** The Unload [`dbunload`] utility has been enhanced (`-ar` option) to allow a single-step unload and reload of a database that can be used whether or not your database is involved in replication.
 ☞ For more information, see “[Unload utility options](#)” [*SQL Anywhere Database Administration Guide*, page 648], and “[Rebuilding databases](#)” [*SQL Anywhere SQL User's Guide*, page 685].
- ◆ **Enhanced transaction log [dblog] output** The Transaction Log [`dblog`] utility now displays additional summary information, including offset information.
 ☞ For more information, see “[Transaction Log utility \(dblog\)](#)” [*SQL Anywhere Database Administration Guide*, page 637].
- ◆ **Log Translation [dbtran] utility enhancements** The Log Translation [`dbtran`] utility permits filtering of the transaction log operations to isolate subsets of operations. This is of particular use to SQL Remote administrators.
 ☞ For more information, see “[Log Translation utility \(dbtran\)](#)” [*SQL Anywhere Database Administration Guide*, page 596].

MobiLink and UltraLite new features

Following is a list of changes and additions to the software since version 6.0.2.

- ◆ **New data types** Real and double data types are now fully supported.

- ◆ **Character set conversion** The MobiLink synchronization server now translates all uploaded characters to Unicode and passes them to the consolidated database using the Unicode ODBC API. Conversely, it translates all downloaded characters from Unicode to the character set of your UltraLite application. Character set conversion within the consolidated database server can influence the results, but the new system allows more consistent behavior across multiple platforms.

☞ For more information, see [“Controlling ODBC driver character set conversion”](#) [*MobiLink Administration Guide*, page 646].

- ◆ **MobiLink synchronization server runs as a Windows NT service** When you run the MobiLink synchronization server as a service, you can configure it to continue running when you log off the Windows NT workstation.

- ◆ **DB2 setup scripts provided** To make it easier to use IBM DB2 as a consolidated database, a DB2 setup script has been added to the available set scripts.

☞ For a list of setup scripts, see [“Setting up a consolidated database”](#) [*MobiLink Administration Guide*, page 6].

Behavior changes in version 6.0.3

This section lists the behavior changes introduced in components of SQL Anywhere Studio version 6.0.3.

Adaptive Server Anywhere behavior changes

- ◆ **Adding columns with default values** When an added column has a default value, the entire column is populated with the default. In previous releases, the column was populated with NULL.
- ◆ **Permissions of referential integrity actions** When changes are made to a primary table, referential integrity actions such as cascading deletes or updates can take place on a secondary table. These actions are implemented using system triggers. The triggers now execute with the permissions of the owner of the secondary table. Previously, they executed with permissions of the owner of the primary table. The new behavior means that cascaded operations can take place between tables with different owners, without additional permissions having to be granted.
- ◆ **datediff, MONTHS, and YEARS functions** The number of months between two dates is now calculated as the number of first-of-the-months between the dates. For example, the difference between January 25 and February 2 is 1; the difference between January 1 and January 31 is 0. The number of years is now calculated as the number of first-of-the-years between the dates.



This changes the results of these functions by one number, in some cases. The change was made for compatibility with Adaptive Server Enterprise.

For the smaller time units there are overflow values to the DATEDIFF function that are now imposed. Previous versions of the software gave incorrect answers if the limit was exceeded.

☞ For a full description, see “[DATEDIFF function \[Date and time\]](#)” [*SQL Anywhere Reference*, page 131].

- ◆ **Default page size** The default page size for databases is now 2048 bytes. This choice is a better choice for many users.
- ◆ **Default database collation** The default collation used when creating databases has changed. The default depends on your operating system settings.

☞ For information on how to find the default collation, see “[Finding the default collation](#)” [*SQL Anywhere Database Administration Guide*, page 307].

- ◆ **SQL Preprocessor default collation** If no collation is explicitly specified, the Embedded SQL Preprocessor now uses locale information to choose a default collation. If the locale information is unavailable, then 850LATIN1 will be used. The collation used is reported following the banner. Previous behavior was to use 850.
 For information on the preprocessor, see “[SQL preprocessor](#)” [[SQL Anywhere Programming Guide, page 430](#)].
- ◆ **Enforced server name length** The server name is checked on startup, and is truncated to a maximum value of 40 characters. On NetBIOS, it is truncated to 16 characters. From the client side, the value of the EngineName parameter is also truncated to 40 characters.
 For more information, see “[EngineName connection parameter \[ENG\]](#)” [[SQL Anywhere Database Administration Guide, page 219](#)].
- ◆ **Agent connection parameter** The Agent connection parameter behavior has been changed. The meaning of this parameter changed from version 5 to version 6, as the need for the *dbclient* executable was removed. The parameter meaning has changed to be more useful in a Version 6 environment.
The Agent connection parameter is deprecated as of version 8.0.1.

SQL Remote behavior changes

The following behavior has changed in SQL Remote version 6.0.3:

- ◆ **Message link parameters stored in the database** By default, the message link parameters are now moved into the database when the Message Agent is run for the first time with the new version of the software. If you have software that explicitly accesses these parameters in their old locations external to the database, it will be affected by this change. You can continue using the old behavior by setting the `external_remote_options` database option to On.
- ◆ **Passwords stored** When a password is entered for a message link, it was not stored in previous versions of the software. As the parameters are now held in the database, a saved password is not held on disk and so is more secure. Passwords are now saved by default. You can continue using the old behavior by setting the `save_remote_passwords` option to OFF.

CHAPTER 13

What's New in Version 6.0.2

About this chapter

This chapter provides an overview to the new features and behavior changes introduced in Adaptive Server Anywhere 6.0.2.

Contents

Topic:	page
New features in version 6.0.2	322
Behavior changes in version 6.0.2	325

New features in version 6.0.2

This section lists the new features introduced in components of SQL Anywhere Studio version 6.0.2.

Adaptive Server Anywhere new features

In addition to bug fixes, Adaptive Server Anywhere version 6.0.2 includes new features in both the software and the documentation.

Cross references

The printed documentation is not necessarily updated with each maintenance release. Cross references in this section may not be valid in the printed documents. For current information, see the online documentation.

- ◆ **UltraLite deployment option** UltraLite databases for small devices such as the PalmPilot and Windows CE computers can be developed with this version of the software.

☞ For information, see the book *UltraLite Developer's Guide*.

- ◆ **Backup and Restore SQL statements** Adding BACKUP and RESTORE as SQL statements provides server side backup and automation of backups using SQL scripts.

The BACKUP statement provides direct backup to tape.

☞ For more information, see “[BACKUP statement](#)” [*SQL Anywhere Reference*, page 318].

- ◆ **Security features** New security features have been added.

- **Auditing** Database administrators can keep track of activity performed on a database by turning on the auditing option. The record of activities is kept in the transaction log. By turning on auditing, you increase the amount of data saved in the transaction log to include login attempts, accurate timestamps of all events, all permissions checks, and all actions requiring DBA authority.

☞ For more information, see “[Auditing database activity](#)” [*SQL Anywhere Database Administration Guide*, page 818].

- **Minimum password length** Database administrators can specify a minimum password length, to discourage easily discovered passwords.

☞ For more information, see “[min_password_length option \[database\]](#)” [*SQL Anywhere Database Administration Guide*, page 418].

- ◆ **Locating servers** A utility is provided for troubleshooting connections.

☞ For more information, see “[The Ping utility](#)” [*SQL Anywhere Database Administration Guide*, page 603].

- ◆ **Starting databases from jConnect connections** Database connections over TDS, including connections from Java applications over jConnect, can start a database on a server.
 - ☞ For more information, see [“Supplying a URL to the driver” \[SQL Anywhere Programming Guide, page 341\]](#).
- ◆ **ODBC 3.51** The ODBC driver has been updated to ODBC 3.51. This version of ODBC includes support for Unicode applications.
 - ☞ For more information, see [“ODBC conformance” \[SQL Anywhere Programming Guide, page 302\]](#).
- ◆ **Control of allowed JOIN syntax** In previous releases, some multi-table queries have been allowed that have ambiguous join clauses. In the present release, you can set an option to disallow such queries.
 - ☞ For more information, see [“extended_join_syntax option \[database\]” \[SQL Anywhere Database Administration Guide, page 395\]](#).
- ◆ **Administration utility enhancements** Options have been added to the administration utilities to provide additional features.
 - **Transaction Log [dbtran] utility** If you use the new -d option, dbtran puts each operation as it occurs in the transaction log file. This makes transaction log output easier to read. This has been added primarily for auditing purposes.
 - ☞ For more information, see [“Log Translation utility \(dbtran\)” \[SQL Anywhere Database Administration Guide, page 596\]](#).
 - **Transaction Log [dbtran] utility** You can run dbtran against a running database server instead of against a log file. This feature has been added to increase the security of the transaction log—there is now no need to access the transaction log directly.
 - ☞ For more information, see [“Log Translation utility \(dbtran\)” \[SQL Anywhere Database Administration Guide, page 596\]](#).
 - **Log Transfer Manager [dbltm] utility logging** New options allow you to tune message logging from these utilities.
 - ☞ For more information, see [“The Log Transfer Manager” \[SQL Anywhere Database Administration Guide, page 589\]](#).
 - **New Log Transfer Manager [dbltm] utility options** New options enable you to replicate only backed up transactions (**backup_only**), and to shut down as soon as all data is replicated (**continuous**).
 - ☞ For more information, see [“The LTM configuration file” \[SQL Anywhere Database Administration Guide, page 592\]](#).

SQL Remote new features

In addition to bug fixes, SQL Remote version 6.0.2 includes the following new features:

- ◆ **Performance enhancements** A major enhancement of the Adaptive Server Anywhere Message Agent (dbremote) operational model for scanning the transaction log and sending messages greatly improves the range of achievable replication turnaround times.

Minimum lag times between entering data at one site and its replication to another site were limited in earlier versions to times on the order of ten minutes. With the new operational model, minimum lag times on the order of seconds can be achieved in some circumstances.

When the Message Agent message-sending process runs in continuous mode, it now stays (**hovers**) at the end of the active transaction log while waiting for more data to be committed, instead of rescanning the transaction log each time. This allows you to poll more frequently, which can significantly reduce time for replication.

☞ For more information, see “[Tuning Message Agent performance](#)” [*SQL Remote User's Guide*, page 150].

- ◆ **SQL Remote message logging** New options allow you to tune message logging from these utilities.

☞ For more information, see “[Message Agent](#)” [*SQL Remote User's Guide*, page 188].

Behavior changes in version 6.0.2

This section lists behavior changes in the components of SQL Anywhere Studio.

Adaptive Server Anywhere behavior changes

The following are behavior changes from previous versions of the software.

- ◆ **Permissions required to debug Java** In order to use the Java debugger, you must either have DBA authority, or be granted membership in the SA_DEBUG group. The SA_DEBUG group does not exist in databases created prior to 6.0.2, and in these older databases any user can use the Java debugger. The SA_DEBUG group was added to close a potential security hole.

☞ For more information, see [“Requirements for using the debugger”](#) [*SQL Anywhere SQL User's Guide*, page 823].

- ◆ **Default packet size change** The default packets size for client/server communications has been changed from 512 bytes to 1000 bytes. This change improves performance for multi-row fetches and fetches of large rows. It also increases the memory requirements.

☞ For more information on packet size, see [“CommBufferSize connection parameter \[CBSIZE\]”](#) [*SQL Anywhere Database Administration Guide*, page 204], and [“-p server option”](#) [*SQL Anywhere Database Administration Guide*, page 162].

CHAPTER 14

What's New in Version 6.0.1

About this chapter

This chapter provides an overview to the new features introduced in Adaptive Server Anywhere 6.0.1

Contents

Topic:	page
New features in version 6.0.1	328
New features in SQL Remote	331
Behavior changes	332

New features in version 6.0.1

This section introduces the new features in Adaptive Server Anywhere version 6.0.1. It provides a listing of major new features, with cross references to locations where each feature is discussed in detail.

Adaptive Server Anywhere for Windows CE

The Microsoft Windows CE operating system developed for handheld computing devices and embedded devices custom-built to carry out a specific task.

Starting with Version 6.0.1, Adaptive Server Anywhere is available for Windows CE. The Windows CE version of Adaptive Server Anywhere has the following characteristics:

- ◆ **Full-featured database** All SQL features in other versions of Adaptive Server Anywhere are available in the Windows CE version, including transaction processing, referential integrity actions, procedures and triggers, and so on.

The Java features and the remote data access features are not available in Windows CE.

- ◆ **Administer from your desktop** When running Windows CE on a device that can be attached to a network or directly to a PC, you can administer your Windows CE database from a Sybase Central running on the PC.
- ◆ **ODBC and Embedded SQL applications** You can use either of these interfaces to develop client applications.
- ◆ **SQL Remote replication** The SQL Remote file link is implemented to be compatible with Windows CE ActiveSync synchronization.

Remote data access

Remote data access gives you access to data on external data sources, as if they were stored on the local database.

☞ For information about remote data access, see the chapter “[Accessing Remote Data](#)” [*SQL Anywhere SQL User's Guide*, page 703] and the chapter “[Server Classes for Remote Data Access](#)” [*SQL Anywhere SQL User's Guide*, page 737].

Character set conversion

Character set conversion has been added to translate strings automatically

between different character sets as data is passed between client applications and the database server. This enables more flexibility in mixed character-set environments.

Character set conversion can be carried out among character sets that represent the same characters, but at different values. There needs to be a degree of compatibility between the character sets for this to be possible. For example, character set conversion is possible between EUC-JIS and Shift-JIS character sets, but not between EUC-JIS and OEM code page 850.

To enable character-set conversion, you must start the database server using the new `-ct` option.

Most of the character set conversion features occur automatically, with little user intervention required.

☞ For a description of character set conversion, see “[Understanding character set conversion](#)” [*SQL Anywhere Database Administration Guide*, page 305].

New Java features

There are some changes made to the Java support. These include the following:

- ◆ **Compressed jar files** You can now install compressed jar files and zip files into the database. However, you should not use the *jar* utility that comes with the Sun JDK. Other zip utilities do produce suitable files.
- ◆ **Result sets from Java procedures** You can wrap Java methods in a stored procedure, which can return a result set or multiple result sets to the calling environment.

☞ For information on this feature, see “[Returning result sets from Java methods](#)” [*SQL Anywhere Programming Guide*, page 98].

- ◆ **Default internal connection** When a database connection is established for internal JDBC operations, it is now recommended that you use the following URL:






```
jdbc:default:connection
```

In version 6.0.0, an empty string was used to establish this connection. While the empty string does still work, it is deprecated. The new URL corresponds to the SQLJ1 proposed standard.

Additional new features

Several other features have been added to Adaptive Server Anywhere 6.0.1.

These include the following:

- ◆ **jConnect 4.0** The version of jConnect included in this product has been updated to version 4.0.
- ◆ **AutoStart connection parameter** This parameter prevents a personal server from starting if no network connection is successful.
 For a description, see [“AutoStart connection parameter \[ASTART\]”](#) [*SQL Anywhere Database Administration Guide*, page 203].
- ◆ **MESSAGE statement** Extensions to the MESSAGE statement allow messages to be directed to the client, the Server Messages window, or a log file.
 For a description, see [“MESSAGE statement”](#) [*SQL Anywhere Reference*, page 570].
- ◆ **Message callbacks** Windows Embedded SQL applications can handle messages received from the server while a request is being processed by registering a message callback function.
 For more information, see [“Implementing request management”](#) [*SQL Anywhere Programming Guide*, page 428].
- ◆ **More control over operating system threads** A new database server option (-gx) controls the number of operating system threads that are in use. The existing -gt option controls how many can be in use at one time, effectively controlling the number of CPUs that can be exploited.
 For more information, see [“The SQL Anywhere database server”](#) [*SQL Anywhere Database Administration Guide*, page 120].
- ◆ **Connection property system procedures** The sa_conn_properties_by_conn and sa_conn_properties_by_name system procedures provide alternative ways of querying connection information.
- ◆ **NULLIF function** This provides an abbreviated form of the CASE expression. NULLIF compares the values of the two expressions. If the first expression equals the second expression, NULLIF returns NULL. If the first expression does not equal the second expression, NULLIF returns the first expression. The NULLIF function provides a short way to write some CASE expressions.
 For more information, see [“Miscellaneous functions”](#) [*SQL Anywhere Reference*, page 94].

New features in SQL Remote

Several features have been added to SQL Remote.

- ◆ **Minimized Message Agent** The Message Agent can be made to start with a minimized window using the -q option.
- ◆ **Message Agent request to resend messages** The point at which the Message Agent requests that a missing message be resent is now user-configurable using the -rp option.
 - ☞ For information on these options, see “[Message Agent](#)” [*SQL Remote User's Guide*, page 188] and “[Tuning incoming message polling](#)” [*SQL Remote User's Guide*, page 151].
- ◆ **Cleaning the stable queue** For Adaptive Server Enterprise, the new -fq option on the Message Agent assists administration by cleaning confirmed messages from the stable queue.
 - ☞ For information, see “[Message Agent](#)” [*SQL Remote User's Guide*, page 188].

Behavior changes

This section describes behavior changes between version 6.0.0 and 6.0.1.

Java system table changes The system tables used to record Java class information (SYSJAR, SYSJARCOMPONENT, and SYSJAVACLASS) had SMALLINT primary keys. These data types have been altered to use INTEGER primary keys. This change allows more Java classes to be stored in a database, and more changes to the Java classes in the database.

This change takes effect for new databases and databases upgraded using the Upgrade utility (dbupgrad) from this or future releases.

Default ansinull setting for Transact-SQL and jConnect connections

This has been changed to On, which matches Adaptive Server Enterprise default behavior.

Database server -v option Prior to Version 6, this option produced verbose output to the transaction logs. This is obsolete, and -v is now used to supply version information.

Database server -gss option The behavior of the -gs server option, used to set the stack size, was complicated. The -gs option is now deprecated, and -gss provides the same functionality in a clearer way.

☞ For more information, see [“The SQL Anywhere database server” \[SQL Anywhere Database Administration Guide, page 120\]](#).

Character set conversion in Interactive SQL Formerly, when the char_oem_translation option was set to DETECT, Interactive SQL would fetch the collation label from the database to determine whether or not OEM to ANSI character set conversion would be turned on. If the collation label started with a string that indicated an ANSI code page, conversion would be turned off. Otherwise it would be turned on. When the option was set to DETECT, Interactive SQL would display a message in the status window indicating the collation label of the database and the display conversion setting.

The new behavior is as follows. If the option is set to DETECT, Interactive SQL will obtain the CharSet connection property from the server. This is the character set that the server is using for sending all character strings on this connection. If this character set indicates an ANSI code page, then OEM to ANSI conversion is turned off. Otherwise it is turned on. A new message is displayed, indicating the collation label of the database, the character set used for communication over this connection, and the display conversion setting.

CHAPTER 15

Upgrading to SQL Anywhere 10

About this chapter

This chapter describes how to upgrade your software and databases.

Contents

Topic:	page
Upgrading SQL Anywhere	334
Upgrading MobiLink	342
Upgrading QAnywhere	348
Upgrading UltraLite	349
Upgrading SQL Remote	359

Upgrading SQL Anywhere

Note

Adaptive Server Anywhere has been renamed SQL Anywhere. In this chapter, SQL Anywhere is used to refer to all versions.

In version 10.0.0, the only way to upgrade your database file format is to rebuild the database, which consists of unloading the old database, and reloading it into a new version 10 database. Attempting to load version 9 or earlier databases results in an error on database startup.

There are three approaches for rebuilding existing databases:

- ◆ Use the version 10.0.0 Unload utility (dbunload) with the -an (create a new database) or -ar (replace the old database) option. See [“Rebuilding a database using the Unload utility” on page 339](#).
- ◆ Use the Upgrade Database wizard in Sybase Central. You can choose to create a new database or replace the old database. See [“Rebuilding a database from Sybase Central” on page 337](#).
- ◆ Unload the database using an older version of dbunload, and then reload the database using the *reload.sql* file and the version 10.0.0 database server. If you need to make schema changes, this is the recommended way of upgrading. After you make the schema changes, you can initialize a new database, and then apply the rebuild script to it.

If you want to change the characteristics of the database during unload and reload (for example, change a case-sensitive database to a case-insensitive database), the procedure is more involved. For more information, see [“Rebuilding databases” \[SQL Anywhere SQL User’s Guide, page 685\]](#).

Compatibility with existing software

- ◆ SQL Anywhere 10 database servers support connections from client applications using software from version 6.0.0 or later.
- ◆ Management of old databases and old database servers from the current version of Sybase Central is provided as follows:
 - Full management of version 8 and later databases running on version 8 and later servers.
 - You can connect to a version 5, 6, 7, 8, or 9 database on a version 8 or later database server to upgrade the database file format by using the Unload Database wizard.
 - There is no support for version 6 and earlier databases running on version 8 and older database servers.

Upgrade quick start

For previous users of the software, the following steps summarize the process for upgrading your database to version 10.

❖ To upgrade a database (command line)

1. Back up the database. For example:

```
dbbackup -c "DBF=mydb.db;UID=DBA;PWD=sql" old-db-backup-dir
```

2. If possible, defragment the drive where the new database will be stored because a fragmented drive can decrease database performance.
3. Shut down all SQL Anywhere and Adaptive Server Anywhere database servers. For example:

```
dbstop
```

4. Unload and reload the old database into a new version 10 database. For example:

```
dbunload -c "DBF=mydb.db;UID=DBA;PWD=sql" -o dbunload_log_
mydb.txt -an mydb10.db
```

5. Shut down the new database and back it up before using it. For example:

```
dbstop -c "DBF=mydb.db;UID=DBA;PWD=sql"
```

```
dbbackup -c "DBF=mydb.db;UID=DBA;PWD=sql" new-db-backup-dir
```

See also

- ◆ [“Rebuilding a database using the Unload utility” on page 339](#)
- ◆ [“Rebuilding a database from Sybase Central” on page 337](#)

Important upgrade precautions

There are several precautions you should take before upgrading any application, and these apply to SQL Anywhere upgrades just as to any other software.

- ◆ **Check the behavior changes** Confirm that none of the documented behavior changes in version 10.0.0 affect your application. If they do, you must update your application accordingly. See [“What’s New in Version 10.0.0” on page 1](#).
- ◆ **Test your application** Test your application thoroughly in a SQL Anywhere 10 environment before upgrading any applications in production use.

- ◆ **Validate and back up the database** Validate your database, and back up your existing software and database. In addition, as recovery cannot happen across a database upgrade, make a backup after upgrading to ensure recoverability going forward.
- ◆ **Synchronize before upgrading** For databases involved in synchronization, such as UltraLite databases or SQL Anywhere remote databases in MobiLink installations, you must perform a successful synchronization before upgrading.
- ◆ **Test your upgrade procedure** Test your upgrade procedure carefully before carrying it out.

SQL Anywhere is used in many different configurations, and no upgrade guidelines can be guaranteed for all cases.

Rebuilding your database for version 10.0.0

This section describes how to unload and reload your database into a new version 10 database.

Caution

Unloading and reloading a large database can be time consuming and can require a large amount of disk space. The process requires access to disk space approximately twice the size of your database to hold the unloaded data and the new database file.

Upgrade restrictions

There are some restrictions to note when rebuilding version 9.0.2 or earlier databases using the 10.0.0 tools:

- ◆ You must disconnect the database from any earlier versions of the database server, and you must shut down any earlier database servers running on the computer. You must also shut down any version 10 database servers that are running on the computer. If dbunload detects any of these cases, it issues an error and fails.
- ◆ Do not include the ENG, START, or LINKS connection parameters in the dbunload connection string for the old database (specified in the -c option). If you specify these parameters, they are ignored and a warning is displayed. On the Connect dialog in Sybase Central, do not enter values in the Server Name and Start Line fields.
- ◆ You must run dbunload on the computer where the old database is located (dbunload *must* be able to connect to the database using shared memory).
- ◆ You cannot run a database server named dbunload_support_engine on the computer where the rebuild is taking place.

- ◆ If you use the Unload utility (dbunload) to rebuild your database, you must set the SATMP and ASTMP environment variables to the same value, or do not set them at all.
- ◆ If you are using NetWare, you must rebuild the database on a Windows or Unix computer. You can then connect to the new version 10.0.0 database on a database server running on the NetWare computer.

Special considerations

- ◆ **Password case sensitivity** In newly-created SQL Anywhere 10 databases, all passwords are case-sensitive, regardless of the case-sensitivity of the database. The default DBA password for new databases is **sql**.

When you rebuild an existing database, SQL Anywhere determines the case sensitivity of the password as follows:

- If the password was originally entered in a case-insensitive database, the password remains case-insensitive.
 - If the password was originally entered in a case-sensitive database, uppercase and mixed case passwords remain case-sensitive. However, if the password was entered in all lowercase, then the password becomes *case-insensitive*.
 - Changes to both existing passwords and new passwords are case-sensitive.
- ◆ **Page sizes** The default database page size for SQL Anywhere 10 databases has been changed to 4096 bytes from 2048 bytes. The only supported page sizes in version 10 are 2048 bytes, 4096 bytes, and 8192 bytes. If your old database uses an unsupported page size, the new database will have a page size of 4096 bytes by default. You can use the -p option to specify a different page size.
 - ◆ **Collations** Unless you specify a new or different collation for the rebuilt database, the collation from the old database will be unloaded and reused in the rebuilt database.

Rebuilding a database from Sybase Central

You can use the Unload Database wizard to rebuild an old database. The wizard lets you choose whether you want to create a new database or replace the old database with a new version 10 database. It is strongly recommended that you back up your database before rebuilding it, especially if you choose to replace the old database with a new one.

❖ **To upgrade the database file format (Sybase Central)**

1. Carry out the standard precautions for upgrading software. See [“Important upgrade precautions” on page 335](#).
2. If possible, defragment the drive where the new database will be stored because a fragmented drive can decrease database performance.
3. Start your database on the old version of the database server.

For example, the following command starts a version 9 database running on a version 9 database server:

```
dbeng9 mydatabase.db
```

4. Ensure that you have exclusive access to the database to be unloaded and reloaded. No other users can be connected.
5. Start Sybase Central.

From the Start menu, choose Programs ► SQL Anywhere 10 ► Sybase Central.

6. Connect to the database you want to rebuild.
7. Right-click the database and choose Unload Database from the popup menu.
The Unload Database wizard appears.
8. Read the introductory page of the wizard, and then click Next.
9. Choose to unload the database to which you are connected. Click Next.
10. Select Unload and Reload into a New Database. Click Next.
11. Specify a new file name for the database. Click Next.

You can specify the page size for the new database. In version 10, the default (and recommended) page size is 4096 bytes.

You can encrypt the database file if you want. You need the encryption key each time you want to start the database. See [“Encrypting a database” \[SQL Anywhere Database Administration Guide, page 825\]](#).

12. Choose to unload structure and data. Click Next.
13. Choose to unload all database objects. Click Next.
14. Specify whether you want to connect to the new database when the unload/reload is complete.

15. Click Finish to start the process. You will need to examine the new database to confirm that the upgrade completed properly.

Rebuilding a database using the Unload utility

You can use the Unload utility (dbunload) -an or -ar option to rebuild an old database. *The -an option is recommended because it creates a new database*, while -ar replaces your old database with a new version 10 database. If you use -ar, it is strongly recommended that you back up your database before rebuilding it.

❖ To upgrade the database file format (command line)

1. Carry out the standard precautions for upgrading software. See [“Important upgrade precautions” on page 335](#).
2. If possible, defragment the drive where the new database will be stored because a fragmented drive can decrease database performance.
3. Ensure that you have exclusive access to the database to be unloaded and reloaded. No other users can be connected.
4. Ensure that the version 10 utilities are ahead of other utilities in your system path. See [“Using the utilities” on page 341](#).
5. Execute the Unload utility (dbunload) using the -an or -ar option to create a new database.

```
dbunload -c "connection-string" -an database-filename
```

For example:

```
dbunload -c "DBF=mydb.db;UID=DBA;PWD=sql" -o dbunload_log_
mydb.txt -an mydb10.db
```

The *connection-string* must connect to the database to be unloaded with DBA authority. This command creates a new database (by specifying -an). If you specify the -ar option, the existing database is replaced with an upgraded database. To use the -ar option, you must connect to a personal server or to a network server on the same computer as the Unload utility (dbunload).

☞ For information on other Unload utility (dbunload) options, see [“Unload utility \(dbunload\)” \[SQL Anywhere Database Administration Guide, page 644\]](#).

6. Shut down the database and back up the transaction log before using the reloaded database.

Known issues

If the rebuild process fails when you run dbunload or the Unload Database wizard, you can use the following steps to help diagnose the reason for the failure.

❖ To diagnose a rebuild failure

1. Run dbunload -n on your old database.

```
dbunload -c "connection-string" -n directory-name
```

2. Create a new, empty version 10 database.

```
dbinit test.db
```

3. Apply the reload.sql file to the empty database.

```
dbisql -c "DBN=test.db;UID=DBA;pwd=sql" reload.sql
```

4. Make changes to the reload.sql file or the original database based on the messages you receive when applying the reload.sql file to the new database.

The following table lists issues that are known to cause a rebuild to fail, as well as their solutions.

Known problem	Solution
<p>A DECLARE LOCAL TEMPORARY TABLE statement in a procedure or trigger causes a syntax error if the table name is prefixed with an owner name.</p> <p>If a CREATE TRIGGER statement does not include an owner name for the table on which the trigger is defined, and the table must be qualified with an owner when referenced by the user executing the reload.sql file, the statement fails with a Table 'table-name' not found error.</p> <p>If an object name corresponds to a reserved word in version 10, such as NCHAR, then the reload fails. For example:</p> <pre>CREATE PROCEDURE p() BEGIN DECLARE NCHAR INT; SET NCHAR = 1; END</pre>	<p>Remove the owner name.</p> <p>Prefix the table name with the owner name.</p> <p>Change all references to the reserved word to use a different name. For variable names, prefixing the name with @ is a common convention that prevents naming conflicts.</p> <p>☞ For a complete list of reserved words, see “Reserved words” [SQL Anywhere Reference, page 4].</p>

Using the utilities

If you have multiple versions of SQL Anywhere on your computer, you must pay attention to your system path when using utilities. Since the installation adds the most recently installed version executable directory to the end of your system path, it is possible to install a new version of the software, and still inadvertently be running the previously installed version.

For example, if an Adaptive Server Anywhere version 8 executable directory is ahead of the SQL Anywhere 10 executable directory in your path and you use the dbinit command, you will use the version 8 utility, and consequently create a version 8 database.

There are three ways you can ensure that you are using the version 10 utilities:

- ◆ Modify your system path so that the SQL Anywhere 10 executable directory is before any previous version executable directory.
- ◆ Change to the SQL Anywhere 10 executable directory before executing your command.
- ◆ Uninstall the old software.




Upgrading MobiLink

Compatibility with existing software


- ◆ New MobiLink clients are incompatible with older versions of the MobiLink synchronization server.
- ◆ The new MobiLink synchronization server can be used with older clients, or with a mixture of clients who are version 10 and previous versions. To use older clients, start the MobiLink server with the -xo option.
- ◆ Version 10 of the MobiLink synchronization server only supports version 8 and 9 SQL Anywhere and UltraLite clients. If you need to support earlier clients, you should keep an older version of the MobiLink synchronization server for supporting them.
- ◆ Confirm that none of the documented behavior changes in version 10.0.0 affect your application. If they do, you must update your application accordingly. See [“What’s New in Version 10.0.0” on page 1](#).

Upgrade order

If you are upgrading an existing MobiLink installation, you must upgrade the components in the following order:

1. Shut down the MobiLink synchronization server.
2. Upgrade the consolidated database.
 See [“Upgrading your consolidated database” on page 343](#).
3. Upgrade the MobiLink synchronization server.
 See [“Upgrading the MobiLink synchronization server” on page 344](#).
4. Start the MobiLink synchronization server.
5. Upgrade the MobiLink clients.
 For information on SQL Anywhere remote databases, see [“Upgrading SQL Anywhere MobiLink clients” on page 345](#). For information on UltraLite applications, see [“Porting UltraLite application code to version 10” on page 355](#).

Before upgrading, check for behavior changes that may affect you and carry out standard upgrade precautions.

 For more information, see:

- ◆ [“Behavior changes and deprecated features” on page 80](#)
- ◆ [“Important upgrade precautions” on page 335](#)

Upgrading your consolidated database

Before you can use the new MobiLink synchronization server with a pre-existing consolidated database, you must run upgrade scripts that install new system objects.

❖ To upgrade a consolidated database (not DB2)

1. If you upgrading a SQL Anywhere consolidated database, you must first upgrade the database to version 10:
 - a. Shut down the database server.
 - b. Upgrade the database to version 10.
 - ☞ For instructions, see [“Upgrading SQL Anywhere” on page 334](#).
 - c. Start the database server, logging in as DBA.

Note: You must log in as DBA to upgrade.
2. If you are upgrading from version 6.x, run the MobiLink setup script, located in the *MobiLink\setup* subdirectory of your SQL Anywhere installation. Do not run the setup script if you are upgrading from a later version.

☞ For more information about setup scripts, see [“MobiLink Consolidated Databases” \[MobiLink Administration Guide, page 3\]](#).

3. Run the appropriate upgrade script for the version you are upgrading from. The upgrade scripts are called *upgrade_XXX.sql*, where XXX indicates your consolidated database. They are located under your SQL Anywhere installation in *MobiLink\upgrade\version*, where *version* is the version you are upgrading from.

For example, to upgrade a SQL Anywhere version 9.0.2 consolidated database, connect to the database in Interactive SQL and run the following command:

```
READ 'c:\Program Files\Sybase\SQL Anywhere 10\MobiLink\
upgrade\902\upgrade_asa.sql'
```

4. If you want a user other than DBA to be able to start mlsrv10, you must grant permission to that user on the new MobiLink system objects:

MobiLink:

- ◆ ml_column MobiLink system table
- ◆ ml_database MobiLink system table
- ◆ ml_add_user system procedure
- ◆ ml_delete_user system procedure

- ◆ ml_reset_sync_state system procedure
- ◆ ml_delete_sync_state system procedure
- ◆ ml_delete_sync_state_before system procedure


Server-initiated synchronization:

- ◆ ml_sis_sync_state MobiLink system table

QAnywhere:


- ◆ ml_qa_clients MobiLink system table

❖ **To upgrade a DB2 consolidated database**

1. Copy the *MobiLink\setup\SyncDB2Long.class* file to the *SQLLIB\FUNCTION* directory on the DB2 server computer. You probably need to restart the instance. For details, see your DB2 documentation.
2. If you are upgrading from MobiLink version 6, create the MobiLink system tables and stored procedures by running the setup SQL script *MobiLink\setup\syncdb2long.sql*.
 For information about how to run the DB2 setup script, see [“IBM DB2 UDB consolidated database” \[MobiLink Administration Guide, page 14\]](#).
3. Locate the DB2 upgrade script.
The upgrade script is called *upgrade_db2tolong.sql* and is held in the *MobiLink/upgrade/version* subdirectory of your SQL Anywhere installation. The *version* directory refers to the version of MobiLink from which you are upgrading.
4. Copy *upgrade_db2tolong.sql* and modify the copy. Change the CONNECT statement at the start of the script so it will work with the instance you want to connect to. Apply the copied SQL script to the consolidated database.

Upgrading the MobiLink synchronization server

Before using the version 10 MobiLink synchronization server, check the behavior changes to see if any affect you.

 For a list of behavior changes, see [“Behavior changes and deprecated features” on page 80](#).

Version 10 of the MobiLink synchronization server only supports version 8 and 9 SQL Anywhere and UltraLite clients. If you need to support earlier clients, you should keep an older version of the MobiLink synchronization server for supporting them.

Upgrading SQL Anywhere MobiLink clients

In a production environment, only upgrade SQL Anywhere remote databases after you have upgraded both the consolidated database and the MobiLink synchronization server.

Note: In version 10.0.0, Adaptive Server Anywhere was renamed to SQL Anywhere.

There are several kinds of upgrade to consider:

- ◆ Upgrading the software.
- ◆ Upgrading the remote database itself.
- ◆ Upgrading the whole application.

Caution

You must complete a successful synchronization just before you update a database involved in MobiLink synchronization. You should also validate and back up the database.

Upgrading the software

It is recommended that you upgrade the dbmlsync MobiLink client and the SQL Anywhere database server at the same time. You must upgrade the remote database before running the new dbmlsync utility.

Version 10 MobiLink clients require a MobiLink version 10 synchronization server for synchronization. Version 10 MobiLink clients do not synchronize with a MobiLink synchronization server earlier than version 10.

For information about upgrading MobiLink, see [“Upgrading MobiLink” on page 342](#).

Upgrading the remote database

You can upgrade MobiLink SQL Anywhere remote databases as described for SQL Anywhere databases. For instructions, see [“Upgrading SQL Anywhere” on page 334](#).

In some cases, such as when there is a schema change or other significant database change, you may need to perform a manual unload and reload.

❖ **To manually unload/reload a remote SQL Anywhere database**

1. Perform a successful synchronization and validate and back up the remote database.
2. Run the dbtran utility to display the starting offset and ending offset of the database transaction log. Make note of the ending offset.

See “The Log Translation utility” [*SQL Anywhere Database Administration Guide*, page 596].

3. Rename the transaction log. This ensures that it is not modified during the unload process. Move the renamed log file to a secure location, such as an offline directory.

4. Unload the database.

☞ See “Rebuilding your database for version 10.0.0” on page 336.

5. Initialize a new database.

☞ See “The Initialization utility” [*SQL Anywhere Database Administration Guide*, page 573].

6. Reload the data into the new database.

☞ See “Rebuilding your database for version 10.0.0” on page 336.

7. Shut down the new database.

8. Erase the new database’s transaction log.

9. Run dblog on the new database, using the following options:

- ◆ Use -z to specify the ending offset that you noted in Step 1.
- ◆ Use -x to set the relative offset to zero.

For example:

```
dblog -x 0 -z 137829 database-name.db
```

☞ See “The Transaction Log utility” [*SQL Anywhere Database Administration Guide*, page 637].

10. Start dbmlsync, specifying the location of the original log file that you moved in Step 2.

11. When you no longer need the old log file, set the database option delete_old_logs.

See “delete_old_logs option [MobiLink] [SQL Remote] [Replication Agent]” [*SQL Anywhere Database Administration Guide*, page 394].

Upgrading applications

When deploying a new version of a MobiLink application, it is recommended that you use a new version name for the synchronization scripts. For example, if the existing application uses a script version called **v1**, then the upgraded application could use a script version called **v2**. Both script versions can be in use at the same time. This makes it easier to upgrade the remote databases incrementally, rather than all at once.

For versions 9.0.0 and later, the MobiLink synchronization server `-zd` option has been removed. If your deployment uses the `-zd` option and you want to upgrade, you must change your download scripts to accept the last download timestamp as the first parameter.

Upgrading QAnywhere

To upgrade a QAnywhere application, you can upgrade your consolidated database, application, and client message stores.

☞ To upgrade the consolidated database, see [“Upgrading your consolidated database” on page 343](#).

To upgrade your applications, you should review the new features and behavior changes in this release.

☞ See [“What’s New in Version 10.0.0” on page 1](#).

❖ To upgrade QAnywhere message stores

1. Deploy QAnywhere files.

☞ See [“Deploying QAnywhere applications” \[MobiLink Administration Guide, page 663\]](#).

2. Upgrade the message store:

Run the QAnywhere Agent with the -su option or the -sur option. See:

- ◆ [“-su option” \[QAnywhere User’s Guide, page 147\]](#)
- ◆ [“-sur option” \[QAnywhere User’s Guide, page 148\]](#)

Upgrading UltraLite

Upgrading previous versions of UltraLite requires that you consider which upgrade path is required for your database and application code.

Compatibility with
existing software

- ◆ The UltraLite 10 runtime and the UltraLite 10 engine do not work with database files and application code created with older versions of UltraLite.
- ◆ UltraLite 10 will not support connections from client applications from any other version of UltraLite except UltraLite 10.
- ◆ Management of old databases and client applications from the current version of Sybase Central is provided as follows:
 - Full management of version 10 databases only. You cannot manage databases from previous versions.
 - You can only connect to a version 5, 6, 7, 8, or 9 database to upgrade the database file format.
 - You can port a version 5, 6, 7, 8, or 9 C/C++ client application to upgrade the source files for these applications with the Migrate C/C++ API wizard.

Using UltraLite 10 utilities

If you have multiple versions of SQL Anywhere on your computer, you must pay attention to your system path when using UltraLite utilities. See [“Using the utilities” on page 341](#).

Upgrading your UltraLite database

Upgrading previous versions of UltraLite databases, requires that you:

- ◆ Synchronize the data.
- ◆ Disconnect all applications and administration tools.
- ◆ Copy the database to your desktop computer.

Note

You cannot connect to an older version of the database with any UltraLite 10 administration tools until you have upgraded the database.

Special database
upgrade considerations

- ◆ The UltraLite schema is now part of the database rather than in a separate .usm file. This means that applications can no longer create a new database in the same way as in earlier versions.
- ◆ File formats have been consolidated in version 10 of UltraLite. This means that most platforms can now share a database and as a result, Unicode characters are no longer required.

If you need characters that are not included your chosen collation, you should now choose to UTF-8 encode your database. See “[Platform requirements for character set encoding](#)” [*UltraLite Database User’s Guide*, page 38] and “[utf8_encoding property](#)” [*UltraLite Database User’s Guide*, page 115].

Windows CE and desktop databases

If you are upgrading databases for either of these platforms and no longer need Unicode characters, do not UTF-8 encode the database. UTF-8 encoding can unnecessarily increase the size of your database.

- ◆ In newly-created version 10 databases, all passwords are case-sensitive, regardless of the case-sensitivity of the database. The default DBA password for new databases is **sql**. Consequently, user IDs, passwords, and even trusted root certificates may not be preserved as you upgrade your database. You must add former users IDs, passwords, and trusted root certificates to the UltraLite 10 database.

UltraLite database upgrade paths

Because database creation approaches can vary in UltraLite, the upgrade process also varies according to the approach taken. The following table captures the approach required depending on what you need to upgrade.

If you need to upgrade a previous version of...	Use this tool...
<ul style="list-style-type: none">◆ A schema file (.usm).◆ A database file (.udb).◆ Palm OS database records (.pdb).	Upgrade Database wizard or Unload Old Database utility (ulunloadold) with Load Database utility (ulload)
An UltraLite database sourced from a SQL Anywhere reference database. ¹	Extract Database wizard or Initialize Database utility (ulinit)

¹Ensure you have upgraded the SQL Anywhere database first. See “[Upgrading SQL Anywhere](#)” on page 334.

☞ For step-by-step upgrade walkthrough using the tools described in this section, see [“UltraLite database upgrade tools” on page 351](#).

UltraLite database upgrade tools

You can upgrade an existing UltraLite database or schema with either the Upgrade Database wizard or the Upgrade Old Database utility (ulunloadold).

- ◆ Choose the wizard if you want to be guided through the process and have help choosing available properties/options.
- ◆ Choose the utility if you have either of the following requirements:
 - You only want to upgrade named tables into a new database.
 - You want to implement a batch-oriented process.

❖ To upgrade an existing UltraLite database to version 10 (Sybase Central)

1. Carry out the standard precautions for upgrading software. See [“Important upgrade precautions” on page 335](#).
2. Start Sybase Central by choosing Start ► Programs ► SQL Anywhere 10 ► Sybase Central.
3. Upgrade your database by choosing Tools ► UltraLite ► Upgrade Database.

The Upgrade Database wizard appears. Before continuing ensure you have decided:

- ◆ What the source is. You can choose either a database or a schema file.
- ◆ Where to output the upgraded database. You can choose from either:
 - **A new UltraLite 10 database** Choose this option to create the database and connect to it.
 - **An existing UltraLite 10 database** Choose this option to change some database options or the collation using settings from the SQL Anywhere reference database. Ensure you choose a character set and collation appropriate for the data in the existing UltraLite database and its internal schema.
- 4. Choose your upgrade source by selecting the appropriate option:
 - ◆ **An Old Database** Browse for a UltraLite database (*.udb or *.pdb).
 - ◆ **An Old Schema File** Browse for an UltraLite schema file (*.usm).
- 5. Connect to the file you selected and then click Next.
- 6. Choose your output destination:

- ◆ **New Database** You must create a new database file and set the database properties you require. Follow the instructions in the wizard.
 - ◆ **Use an Already Connected Database** You can choose a connected database from the list provided for you.
 - ◆ **Use and Existing Database That You Are Not Connected to** Click Database to open the Connect dialog and connect to the existing UltraLite 10 database.
7. Follow the instructions of the wizard to make additional choices relating to your output destination. If you had trusted root certificates included in your previous version of UltraLite, ensure you add them back into the new UltraLite database.
 8. Click Finish to upgrade the database.
 9. If you had users in the previous version of the UltraLite database, and if you do not see them in the new database you have created, remember to add them. See [“Working with UltraLite users” \[UltraLite Database User’s Guide, page 80\]](#).
- ❖ **To upgrade an existing UltraLite database to version 10 (command line)**
1. Carry out the standard precautions for upgrading software. See [“Important upgrade precautions” on page 335](#).
 2. Ensure that the version 10 UltraLite utilities are ahead of older UltraLite utilities in your system path. See [“Using the utilities” on page 341](#).
 3. Open a command prompt and run the Unload Old Database utility (ulunloadold) to create an XML intermediary file using the following syntax:


```
ulunloadold -c "connection-string" [ options ] xml-file
```

Ensure that you have:

 - ◆ Named the XML file that the ulunloadold utility is to create.
 - ◆ Used either the DBF or schema_file parameter in your *connection-string*, depending on whether or not you want to upgrade an old UltraLite database (*.udb or *.pdb) or an old UltraLite schema file (*.usm).

All other options are discretionary.

☞ For a complete reference for this utility see [“UltraLite Unload Old Database utility \(ulunloadold\)” \[UltraLite Database User’s Guide, page 169\]](#).

4. Execute the Load XML to Database utility (ulload) to reload the XML into a new or existing UltraLite 10 database.

If you are loading the XML into a new database, the **-c** *connection-string* option sets the connection parameters for that database (for example, the UID and PWD required to authenticate the UltraLite user).

The **-o** [*extended-options*] you set depends on whether or not you are changing the characteristics/properties of the database (for example, changing a case-sensitive database to a case-insensitive database).

☞ For a complete reference, see “UltraLite Load XML to Database utility (ulload)” [*UltraLite Database User’s Guide*, page 158].

For example, upgrade an UltraLite 8.x schema file named *dbschema8.usm* into an existing UltraLite version 10 database named *db.udb* with an intermediary XML file named *dbschema.xml* requires these two commands:

```
ulunloadold -c schema_file=dbschema8.usm dbschema.xml
```

```
ulload -c DBF=db.udb dbschema.xml
```

Initialization/extraction
tools

You can extract an UltraLite database from a version 10 SQL Anywhere database with either the Extract Database wizard or the Initialize Database utility (ulinit).

Remember: set reference database properties with UltraLite usage in mind

The UltraLite database is generated with the same property settings as those in the SQL Anywhere reference database. By setting these options in the reference database, you also control the behavior of your UltraLite database.

- ◆ Choose the wizard if you want to be guided through the process and have help choosing available properties/options.
- ◆ Choose the utility if you have either of the following requirements:
 - You only want to upgrade named tables into a new database.
 - You want to implement a batch-oriented process.

❖ **To initialize/extract an UltraLite database from a SQL Anywhere reference (Sybase Central)**

1. Carry out the standard precautions for upgrading software. See “Important upgrade precautions” on page 335.

2. Ensure you have already upgraded your existing SQL Anywhere database and prepared it with UltraLite usage in mind. If you need to update publications, be sure to do that before re-creating the UltraLite database.

☞ For SQL Anywhere upgrade procedures, see [“Using the utilities” on page 341](#). For details on how to prepare SQL Anywhere for use with UltraLite, see [“Creating an UltraLite database from a SQL Anywhere reference database”](#) [*UltraLite Database User’s Guide*, page 31].

3. Start Sybase Central by choosing Start ► Programs ► SQL Anywhere 10 ► Sybase Central.
4. Extract an UltraLite version of the SQL Anywhere database by choosing Tools ► UltraLite ► Extract Database.

The Extract Database wizard appears.

5. Follow the instructions in the wizard.

❖ To initialize/extract an UltraLite database from a SQL Anywhere reference database (command line)

1. Carry out the standard precautions for upgrading software. See [“Important upgrade precautions” on page 335](#).
2. Ensure that the version 10 UltraLite utilities are ahead of older UltraLite utilities in your system path. See [“Using the utilities” on page 341](#).
3. Ensure you have already upgraded your existing SQL Anywhere database and prepared it with UltraLite usage in mind. If you need to update publications, be sure to do that before re-creating the UltraLite database.

☞ For SQL Anywhere upgrade procedures, see [“Using the utilities” on page 341](#). For details on how to prepare SQL Anywhere for use with UltraLite, see [“Creating an UltraLite database from a SQL Anywhere reference database”](#) [*UltraLite Database User’s Guide*, page 31].

4. Open a command prompt and run the Initialize Database utility (ulinit) to extract an UltraLite database using the following syntax:

```
ulinit -a "SAconnection-string" -c "ULconnection-string"
      -n pubname [ options ]
```

Ensure that you have:

- ◆ Provided connection strings for both the upgraded SQL Anywhere reference database and the new UltraLite database you are creating with this command.
- ◆ Named the publications that contain the tables that your UltraLite database requires. To extract all tables, use **-n***.

All other options are discretionary.

Note

Because you are creating a new database, parameters like **UID** and **PWD** are used to create the initial user ID and password for authentication purposes. The SQL Anywhere database is not referenced in this case. However, to override other SQL Anywhere reference database property defaults use **-o [extended-options]**. See “Supported extended options” [*UltraLite Database User’s Guide*, page 172] for a complete list.

☞ For a complete reference, see “UltraLite Load XML to Database utility (ulload)” [*UltraLite Database User’s Guide*, page 158]. For more information on UltraLite database properties you can configure, see “UltraLite Database Settings Reference” [*UltraLite Database User’s Guide*, page 85].

Porting UltraLite application code to version 10

Porting UltraLite applications requires that you must make changes to your application code.

Before you begin, however, ensure you have closely reviewed “UltraLite new features” on page 138 to see what changes have been made to the API you have employed.

Special application upgrade considerations

- ◆ The UltraLite schema is now integrated into the database as tables, rather than as a separate *.usm* file. This means you cannot create a database on device using this file. Instead, a new database creation function/method has been added. However, this method can increase the size of your application. To avoid application bloat, use an administration tool to create the database on the desktop first and deploy it to your application at a later time. See “Desktop creation” [*UltraLite Database User’s Guide*, page 28].
- ◆ Your connection code needs to be updated. You should try to use the connection parameter control, which assembles a set of connection parameters conveniently. The UltraLite runtime in turn transforms the assembled parameters into a connection string. You can still use a connection object, but the connection parameter control provides better diagnosis of connection string errors. If you can, you may want to reprogram applications to use the connection parameter control.
- ◆ UltraLite always has authentication enabled in this version and can support up to four user IDs and passwords. However, if you do not want to maintain authentication in your database, do not create or supply any user IDs and passwords. UltraLite always supplies the defaults of

UID=DBA and **PWD=sql** if none are supplied. See [“Working with user IDs and passwords”](#) [*UltraLite Database User’s Guide*, page 54].

- ◆ If you have multiple embedded SQL files, you must still preprocess each one of them using the SQL preprocessor (sqlpp) to create your C/C++ source files. However, you no longer need to use a reference database. UltraLite database now support embedded SQL directly.
- ◆ Unicode characters are not supported in the same way as in previous versions. Instead, version 10 UltraLite databases use UTF-8 encoding for multi-byte characters. As a result, you no longer need to plan for a Unicode database running on a non-Unicode runtime.

UltraLite application code upgrade paths

Because development APIs and approaches can vary in UltraLite, the upgrade process also varies according to the approach taken. The following table captures the approach required depending on what you need to upgrade.

If you need to upgrade a previous version of...	Do this...
A ulgen generated C/C++ applica- tion	<div>1. Use the Extract Database wizard or run the UltraLite Initializa- tion utility (ulinit) to create a version 10 UltraLite database. See “Initialization/extraction tools” on page 353.²</div> <div>2. Use the Migrate C/C++ API wizard to read tables and statements from the SQL Anywhere 10 project for UltraLite to migrate the API accordingly. See “UltraLite application code upgrade tools” on page 357.</div>
An embedded SQL application	Changes are trivial. No tool required.
A Java application	Java is not in version 10 of UltraLite. You must re-write your application with a supported API.

²Ensure you have upgraded the SQL Anywhere database first. See [“Upgrading SQL Any-where”](#) on page 334.


If you need to upgrade a previous version of...	Do this...
UltraLite components	<p>Major code rewrites include the following:</p> <ul style="list-style-type: none"> ♦ Schema re-writes Because the schema has been integrated into the database with this version, all components need to rewrite the OpenWithCreate functions for ULConnection objects, as well as to remove all schema upgrade code. Instead, you can create a database on the device with ULDatabaseManager.-CreateDatabase. However, to reduce the amount of code required to define the properties in the new database, you should create a database on desktop and deploy it to the device upon completion. For complete details, see “On-device creation” [<i>UltraLite Database User’s Guide</i>, page 33] and “Desktop creation” [<i>UltraLite Database User’s Guide</i>, page 28]. ♦ Connection re-writes All interfaces that used a connection parameter object have been removed: this includes functions or methods that created, opened or dropped the database. Instead, you can use the string interface to pass in these connection parameters.

UltraLite application code upgrade tools

You can only use the Migrate C/C++ API wizard to help you migrate your ulgen-generated C/C++ source code. This wizard helps you identify embedded SQL that may no longer conform to version 10 specifications. If you can not complete the migration process, you can stop and save the SQL statements that you have modified to a *.uag file.

If this is your first time using the wizard, your table and statement source is the SQL Anywhere reference database. For subsequent iterations, you can use the saved *.uag file instead.

❖ To migrate the UltraLite C/C++ API (Sybase Central)

1. Carry out the standard precautions for upgrading software. See “Important upgrade precautions” on page 335.
2. Ensure you have already upgraded your existing SQL Anywhere database and prepared it with UltraLite usage in mind. If you need to update publications, be sure to do that before re-creating the UltraLite database.
 For the SQL Anywhere upgrade procedures, see “Using the utilities” on page 341. For details on how to prepare SQL Anywhere for use with

UltraLite, see “Creating an UltraLite database from a SQL Anywhere reference database” [*UltraLite Database User’s Guide*, page 31] for details.

3. Ensure you have upgraded an UltraLite database. Otherwise you need to extract an UltraLite database from the SQL Anywhere reference database.

This UltraLite database is be used as the validation database for this process.

4. Start Sybase Central by choosing Start ► Programs ► SQL Anywhere 10 ► Sybase Central.
5. Migrate your C/C++ API application by choosing Tools ► UltraLite ► Migrate C/C++ application.

The Migrate C/C++ API wizard appears.

6. On the SQL Statement Source page, choose the source from which statements and tables are read.
 - ◆ First-time users must click Specify Database Information to choose your SQL Anywhere reference database. If you are already connected to the SQL Anywhere database, you can use the DSN you have created for that database connection session.
 - ◆ Otherwise, you can also open previous changes written to the *.uag file you have created for that purpose.
7. Depending on the source you have selected, follow the instructions in the wizard and validate all SQL statements. Invalid statements have a red × next to the statement name. For a complete reference for each supported statement in UltraLite, see “UltraLite SQL Statement Reference” [*UltraLite Database User’s Guide*, page 281].

To correct invalid SQL statements:

- a. Select an invalid statement.
- b. Correct the statement in the text box provided.
- c. Click Validate All SQL Statements.

If the statement is validated, it is placed at the bottom of the list and has a green checkmark next to the statement name. Click Cancel at any time to save your changes to the .uag file and exit the wizard.

Upgrading SQL Remote

If you are upgrading an existing SQL Remote installation from version 6 or later, you must upgrade each database server before or along with its Message Agent (dbremote). You can upgrade Message Agents in any order.

Version 5 users must follow the instructions in “[Upgrading version 5 SQL Remote installations](#)” on page 359.

- ◆ **Upgrade databases** You must upgrade the database file format by unloading and reloading your database. There is no need for all databases to be upgraded at the same time.

☞ For instructions on unloading and reloading the database, see “[Upgrading SQL Anywhere](#)” on page 334.

- ◆ **Software upgrades can be one site at a time** Older Message Agents can exchange messages with version 10 Message Agents.

- ◆ **Message Agent and database server can be upgraded separately** The database server can be upgraded before the Message Agent. It is, however, recommended that you upgrade your Message Agent at the same time as the database server for performance reasons.

- ◆ **Upgrading Adaptive Server Enterprise consolidated databases** SQL Remote no longer supports Adaptive Server Enterprise consolidated databases. To synchronize Adaptive Server Enterprise databases, you should upgrade to MobiLink.

☞ For information about upgrading from SQL Remote to MobiLink, see http://www.ianywhere.com/whitepapers/migrate_to_ml.html.

Upgrading version 5 SQL Remote installations

SQL Remote installations include a consolidated database and many remote databases, together with a Message Agent at each site.

At each site, the Message Agent handles the sending and receiving of messages. The messages take the form of SQL statements, and the database server handles the actual execution of those SQL statements.

The upgrade requirements for SQL Remote are as follows:

- ◆ **Upgrade your databases** You must upgrade the database file format by unloading and reloading your database.

☞ See “[Upgrading SQL Anywhere](#)” on page 334.

- ◆ **Software upgrades can be one site at a time** Version 5 Message Agents can exchange messages with version 10 Message Agents, as long

as the compression database option is set to a value of -1 (minus one). There is no need to upgrade software throughout the installation simultaneously.

- ◆ **Message Agent and server can be upgraded separately** The Message Agent is an embedded SQL application. Therefore, the database server can be upgraded before the Message Agent as long as the compatibility library is used. It is, however, recommended that you upgrade your Message Agent at the same time as the database server for performance reasons.

The Message Agent cannot be upgraded before the database server, as a new client application cannot work with a version 5 server.

Example

One approach to upgrading is as follows:

1. Upgrade the consolidated database server and Message Agent. Set the compression database option to -1 so that all messages are compatible with the version 5 software at remote sites.
2. Upgrade remote database servers and Message Agents. You can set the compression database option to a value other than -1 to take advantage of compression and encoding on messages being sent to the consolidated database server.
3. When all remote database servers and Message Agents are upgraded, set the compression database option at the consolidated site to a value other than -1.

CHAPTER 16

Upgrading Version 5 Applications

About this chapter

The client/server communication protocol changed between SQL Anywhere version 5 and SQL Anywhere version 10. This chapter provides upgrade instructions for those users upgrading from version 5 to a newer release of the software.

This chapter describes upgrade procedures that apply to the change in communication protocol for users of SQL Anywhere versions 5.0 and 5.5, and users of Watcom SQL 4.0. In particular, it addresses the problem of stepwise upgrades of client/server installations.

Contents

Topic:	page
Understanding version 5 upgrades	362
Review of SQL Anywhere Version 5 architecture	365
Upgrading embedded SQL applications	368
Upgrading ODBC applications	373
Using the compatibility library	378
Capturing dbclient command information	384
Upgrading databases	386
Upgrading version 5 SQL Remote installations	387

Understanding version 5 upgrades

A database application and DBMS consists of several components. If your organization has a large SQL Anywhere installation, you may have many client computers, possibly running several applications, connected to more than one server.

When upgrading your system, you need to decide which components to upgrade, and in which order. This chapter guides you in making those choices and carrying out the upgrade.

SQL Anywhere Version 5

In this chapter, **SQL Anywhere Version 5** refers to both versions 5.0 and 5.5 of SQL Anywhere.

☞ In order to upgrade Version 5 applications, it is helpful to understand how connections work with later versions of the software. For information, see “[Connecting to a Database](#)” [*SQL Anywhere Database Administration Guide*, page 39].

The components in your system

If you are currently running SQL Anywhere Version 5, you will have some or all of the following components in your system:

- ◆ **Application** Your application, aside from the SQL Anywhere components.
- ◆ **Connection parameters** SQL Anywhere Version 5.0 connection parameters. These may be assembled from an ODBC data source, or in some other way.
- ◆ **Driver manager** The ODBC driver manager, for ODBC applications.
- ◆ **ODBC driver** The SQL Anywhere Version 5 ODBC driver, for ODBC applications. For network applications, the ODBC driver is on the client computer.
- ◆ **Interface library** The SQL Anywhere Version 5 interface library is used by ODBC and embedded SQL applications. For network applications, the interface library is on the client computer.
- ◆ **SQL Anywhere client** The *dbclient.exe* executable and its command line, for network applications. The command line may specify the server name, and a set of network protocol options. It may be stored in a batch file or an ODBC data source Start Line parameter. For network applications, the SQL Anywhere client is on the client computer.

- ◆ **The database server** The SQL Anywhere Version 5 database server. For network applications, this may be on a separate computer from the client components.
- ◆ **The database** A SQL Anywhere Version 5 database. This is on the same computer as the database server.

Database upgrades not required

You do not need to upgrade your database in order to use newer database server with an existing application.

Major upgrading issues

The major issues in upgrading from version 5 arise from the change in client/server communication protocol. This change means that the Version 5 interface library is not able to communicate with more recent database servers.

To help with this issue, SQL Anywhere includes a compatibility library. This DLL allows communications to both Version 5 and current database servers.

Behavior changes

You should also check the behavior change lists elsewhere in this book for versions since version 5 for any changes that may affect your application.

In addition, applications using Transact-SQL outer joins should note the following incompatibility with newer releases:

The null-supplying table in a Transact-SQL outer join cannot also participate in another regular or outer join. For example, in the following query, table S violates this limitation as it is the null-supplying table in `R.x *= S.x`, and participates in another join.

```
-- invalid query
SELECT *
FROM R, S, T
WHERE R.x *= S.x
AND S.y = T.y
```

When to upgrade your database

There is no need to upgrade your Version 5 database itself when you start using the current release of the software. You do need to upgrade the database if you want to take advantage of many of the features introduced since Version 5.

☞ For more information on upgrading databases, see [“Upgrading SQL Anywhere” on page 334](#).

Running more than one version of the software

The SQL Anywhere Studio software is designed so that both separate versions can be run if you install them in separate directories. This is the default behavior of the SQL Anywhere Setup program.

Review of SQL Anywhere Version 5 architecture

This section reviews the architecture for SQL Anywhere Version 5 applications connecting to a SQL Anywhere Version 5 database.

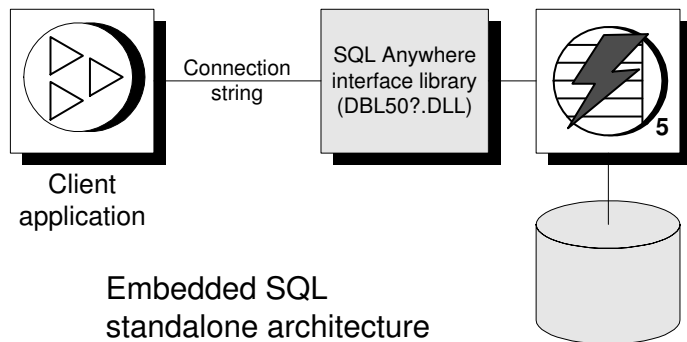
This information helps you to understand the changes needed when upgrading to more recent versions. If you are familiar with SQL Anywhere Version 5 architecture, you do not need to read this section.

Standalone components for Version 5 (embedded SQL)

If you are using SQL Anywhere Version 5 as a personal server, with an embedded SQL client application, you are using the following components on your computer:

- ◆ A SQL Anywhere Version 5 database.
- ◆ The SQL Anywhere Version 5 database engine (personal database server).
- ◆ The SQL Anywhere Version 5 interface library.
- ◆ A SQL Anywhere Version 5.0 connection string.

The following figure illustrates how these pieces fit together.



Here, the question mark in *dbl50?.dll* represents a single character indicating the operating system. The interface library is named *dbl50t.dll* on Windows operating systems.

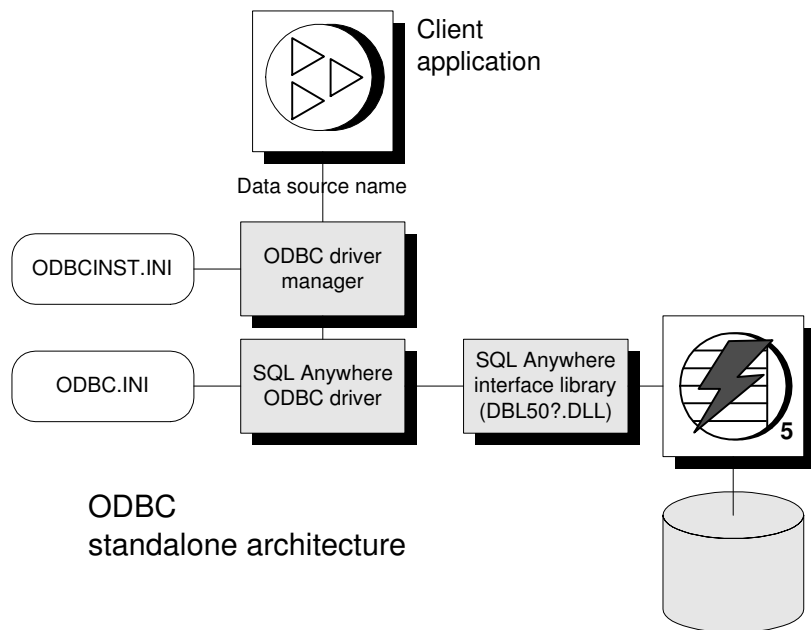
Embedded SQL client applications make calls to the interface library. The interface library is referenced by name, so that the name *dbl50t.dll* is a part of the client application.

Standalone components for Version 5 (ODBC)

If you are using SQL Anywhere Version 5 as a personal server, with an ODBC client application, you are using the following components on your computer:

- ◆ A SQL Anywhere Version 5 database.
- ◆ The SQL Anywhere Version 5 database engine.
- ◆ The SQL Anywhere Version 5 interface library.
- ◆ The SQL Anywhere Version 5 ODBC driver.
- ◆ The ODBC driver manager.
- ◆ A SQL Anywhere Version 5.0 connection description. This may be an ODBC data source, or a connection string from an application.

The following figure illustrates how these pieces fit together. The client application passes a data source name to the ODBC driver manager. The ODBC driver manager looks up the appropriate driver in ODBCINST.INI. The driver looks up the connection information in ODBC.INI and, via the interface library, connects to the SQL Anywhere Version 5 database engine.



Client/server components for Version 5

If you are using SQL Anywhere as a network server, you have the following components on your server computer:

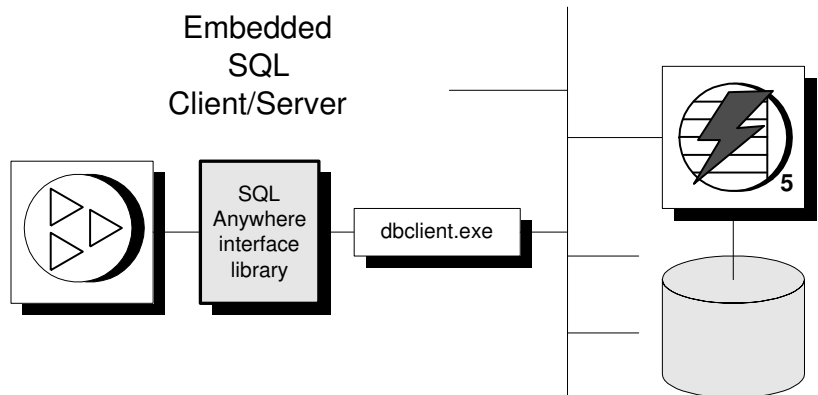
- ◆ A SQL Anywhere Version 5 database
- ◆ The SQL Anywhere Version 5 database server

You have the following components on your client computer:

- ◆ The SQL Anywhere Version 5 Client executable
- ◆ The SQL Anywhere Version 5 interface library.
- ◆ The SQL Anywhere ODBC driver, if your application uses ODBC.
- ◆ A SQL Anywhere Version 5.0 connection description. This may be an ODBC data source, or a connection string from an application.

The data source may contain connection information in the start option, corresponding to a SQL Anywhere Client command line. You may also have connection strings in your application, and batch files that start a client with a particular set of parameters and options.

The architecture of a Version 5 embedded SQL client/server connection is illustrated in the figure. For ODBC applications the ODBC driver manager and ODBC driver stand between the application and the interface library.



Upgrading embedded SQL applications

You can upgrade a SQL Anywhere Version 5 standalone application to use newer versions of the database server by upgrading the database server and the interface library. You do not need to upgrade the database or the client application itself.

The upgrade procedure uses the **compatibility library**. The compatibility library is a dynamic library (a DLL on PCs, shared library on UNIX) that enables embedded SQL applications to work with both Version 5 and newer versions of the database server.

☞ For a description of the compatibility library, see [“Using the compatibility library” on page 378](#).

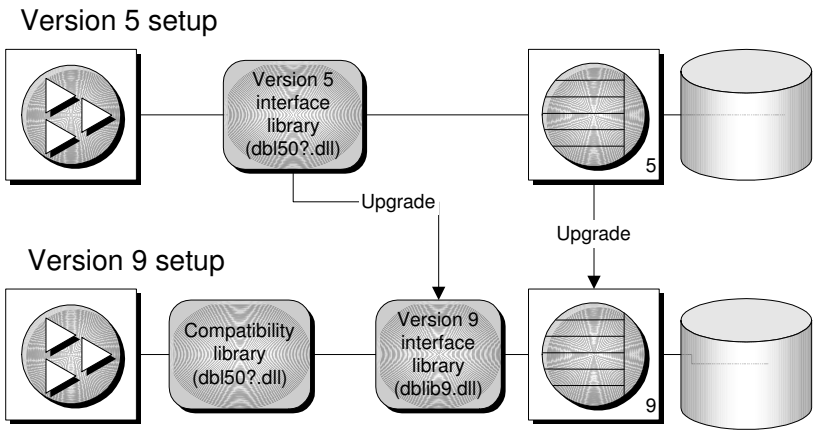
StartLine connection parameter

If your application connection string uses a StartLine parameter that explicitly provides a *dbclient.exe* or *dbeng50.exe* command line, and this is hard coded into your application, there are additional upgrade considerations.

For information on upgrading applications that use StartLine connection parameters, see [“Start parameters and the compatibility library” on page 382](#).

Components upgraded

The following figure illustrates the changes you need to make in your setup in order to upgrade:



☞ For a description of the compatibility library, see [“Using the compatibility library” on page 378](#).

Upgrade procedures for embedded SQL applications

This section provides step-by-step procedures for different kinds of embedded SQL applications:

- ◆ Applications using an embedded database are standalone applications using the personal database server (*dbeng50.exe* for Version 5, *dbeng10.exe* for version 10).
- ◆ Client/server applications connect across a network to the network database server. In Version 5, these applications use the *dbclient.exe* executable.

❖ To upgrade embedded SQL applications that use an embedded database

1. **Install the current version of the software** Newer versions of the software contain components that enable Version 5 applications to continue working. You can either install into a separate directory or over the top of your Version 5 software.

The installation places the new executable directory ahead of the Version 5 executable directory in your system path.

2. **Ensure that your application is using the compatibility DLL** If necessary, copy the compatibility library *db150?.dll* from your SQL Anywhere executable directory to a place where your application will locate it.

For example, you could copy the compatibility library to the same directory as the module of your application that loads it. The file *db150?o.dll* is installed into your SQL Anywhere executable directory. This directory must be in your path as this library is required by the compatibility library.

At this stage, your Version 5 application should continue to work as before. However, it will be connecting to your database through the compatibility library rather than directly through the Version 5 interface library.

☞ If you have any problems at this stage, you need to check how your application locates the interface library. For information, see [“File locations and the compatibility library” on page 381](#).

3. **Create a new connection description** If your application obtains its connection parameters from configuration files, batch files, or the system registry, you should prepare a new description that uses the current version of the database server. For example, Sybase Central stores connection descriptions in the system registry.

- ◆ If you store a connection string that uses the DBF parameter to start the default database server, then the new database server is started automatically by the compatibility library, instead of the Version 5 database server. In this case, no new connection description is needed.
 - ◆ If you store a connection string that uses a Start Line parameter specifying *dbeng50.exe*, you must replace this with one specifying *dbeng10.exe*. If your application contains a hard-coded connection string, you need to take extra steps at this point.
 - ☞ For more information on upgrading StartLine parameters, see [“Start parameters and the compatibility library” on page 382](#).
 - ◆ If the database server is started in some other way, such as by a batch file or using a Windows NT service, you must reconfigure this so that the new version of the database server is started instead.
4. **Use the new connection description** With this step, you are using all current software and have completed your upgrade. The database itself does not need to be upgraded to work with existing applications.

❖ To upgrade embedded SQL client/server applications

1. **Prepare for the upgrade** This step must be carried out at each client computer. You prepare for the upgrade by installing the current version of the software.

Ensure that the installation places the current executable directory ahead of the Version 5 executable directory in your system path.

2. **Ensure that your application is using the compatibility DLL** This step must be carried out at each client computer. If necessary, copy the compatibility library *db150?.dll* from your SQL Anywhere executable directory to a place where your application will locate it.

For example, you could copy the compatibility library to the same directory as the module of your application that loads it. The file *db150?o.dll* is installed into your SQL Anywhere executable directory. This directory must be in your path as this library is required by the compatibility library.

At this stage, your Version 5 application should continue to work as before. However, it will be connecting to your database through the compatibility library rather than directly through the Version 5 interface library.

☞ If you have any problems at this stage, you need to check how your application locates the interface library. For information, see [“File locations and the compatibility library” on page 381](#).

3. **Create a new connection description** This step must be carried out at each client computer. If your application obtains its connection parameters from configuration files, batch files, or the system registry, you should prepare a new description that uses the newer database server. This description is for use when the server is upgraded.

If you store a connection string that uses a StartLine connection parameter specifying *dbclient.exe*, you must replace this with a new one. The new connection description should contain all the dbclient information as a set of parameters. If your application contains a hard-coded connection string, you need to take extra steps at this point.

☞ For more information on upgrading StartLine parameters, see [“Start parameters and the compatibility library” on page 382](#).

☞ For more information about creating connection descriptions that capture the dbclient command information, see [“Capturing dbclient command information” on page 384](#).

4. **Upgrade the database server** This step must be carried out at the server computer.
 - ◆ As with any software upgrade, back up your database before upgrading.
 - ◆ Install SQL Anywhere on the server computer.
 - ◆ Start the new database server on the database.
5. **Use the new connection description** This step must be carried out at each client computer. You need to use the new connection description to connect to the newer version of the database server. With this step, you are using all current software and have completed your upgrade. The database itself does not need to be upgraded to work with existing applications.

Using the version 5 utilities with SQL Anywhere

For the Version 5 database utilities, connection strings are supplied interactively. The Version 5 database utilities such as Interactive SQL are embedded SQL applications that search for the interface library in the following order:

1. The current directory
2. The Version 5 executable directory
3. The system path

For these applications, even though the compatibility library is ahead of the Version 5 interface library in the system path, the Version 5 compatibility library is located.

❖ **To use Version 5 utilities with SQL Anywhere Version 10**

1. Make a backup copy of your interface library file.
2. Copy the compatibility library from your Version 10 executable directory to your Version 5 directory. For example, on Windows XP, copy the file *dbi50t.dll* from the *win32* subdirectory of your Version 10 installation to the *win32* subdirectory of your Version 5 installation.
3. You can now run your Version 5 utilities against both Version 5 and Version 10 database servers.

Upgrading ODBC applications

You can upgrade Version 5 ODBC applications in the following ways:

- ◆ Replace the Version 5 ODBC data source with a current ODBC data source. This approach is a complete upgrade, and is described in this section.
- ◆ Use the compatibility library to connect to a current version of the database server. If you choose this route, your application continues to use the Version 5 ODBC driver, so this is not a complete upgrade. The procedure for upgrading in this way is the same as for embedded SQL applications.

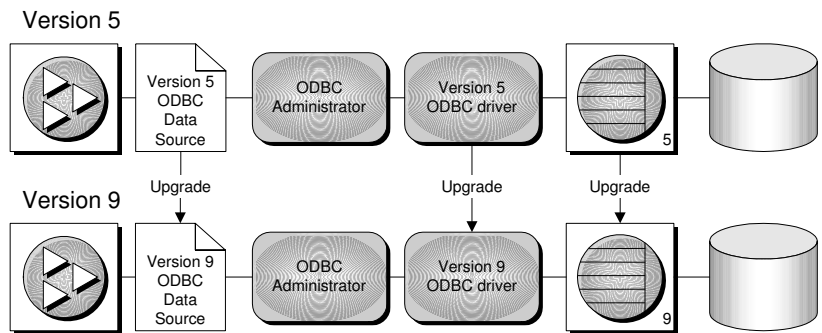
☞ For information about upgrading in this manner, see [“Upgrading embedded SQL applications”](#) on page 368.

StartLine parameter

If your ODBC data source contains a start line specifying the Version 5 standalone engine (*dbeng50.exe*) or the SQL Anywhere client (*dbcclient.exe*), you cannot upgrade using the compatibility library.

Components upgraded

The following figure illustrates the changes you must make when upgrading an ODBC standalone application.



Data sources

The ODBC data source specifies which ODBC driver to use. When an ODBC data source is created as an SQL Anywhere Version 10 data source, it uses the Version 10 ODBC driver.

Data source must be current version

ODBC applications require the version of the ODBC data source to be the same as that of the ODBC driver.

Upgrade procedures for ODBC applications

This section provides step-by-step procedures for different kinds of ODBC applications:

- ◆ Applications using an embedded database are standalone applications using the personal database server (*dbeng50.exe* for Version 5, *dbeng10.exe* for Version 10).
- ◆ Client/server applications connect across a network to the network database server. In Version 5, these applications use the *dbclient.exe* executable.

Some applications allow you to change the ODBC data source name you use. Other applications use a fixed data source name. You can upgrade either kind of application.

❖ To upgrade ODBC applications that use an embedded database

1. **Install the current software** The current version of the software contains components that enable Version 5 ODBC applications to continue working. You can either install into a separate directory or over the top of your Version 5 software.

Your application should be unaffected by installation of the current version of the software.

2. **Create a Version 10 ODBC data source** The changes you make depend on the connection parameters you use in the data source.
 - ◆ If you start the default database server using the DBF parameter, you can use the same connection parameters in your new data source as your old one.
 - ◆ If you store a connection string that uses a Start Line parameter specifying *dbeng50.exe*, you must replace this with one specifying *dbeng10.exe*.
 - ☞ For more information on upgrading StartLine parameters, see [“Start parameters and the compatibility library” on page 382](#).
 - ◆ If the database server is started in some other way, such as by a batch file or using a Windows NT service, you must reconfigure this so that the newer version of the database server is started instead.
 - ☞ For information on creating data sources, see [“Working with ODBC data sources” \[SQL Anywhere Database Administration Guide, page 56\]](#).

3. **Use the new data source** With this step, you are using all current software and have completed your upgrade. The database itself does not need to be upgraded to work with existing applications.

Some applications may have the data source name hard wired. In this case, you need to replace the Version 5 data source with a Version 10 data source of the same name. It is recommended that you rename, rather than delete, your Version 5 data source.

❖ **To upgrade ODBC client/server applications**

1. **Prepare for the upgrade** This step must be carried out at each client computer. You prepare for the upgrade by installing the current version of the software.

Your application should be unaffected by installation of the current version of the software.

2. **Create a Version 10 ODBC data source** This step must be carried out at each client computer. The changes you make depend on the connection parameters you use in the data source.

- ◆ If you start the default database server using the DBF parameter, you can use the same connection parameters in your new data source as your old one.
- ◆ If you store a connection string that uses a StartLine connection parameter specifying *dbclient.exe*, you must replace this with a new one. The new connection description should contain all the *dbclient* information as a set of parameters. If your application contains a hard-coded connection string, you need to take extra steps at this point.
 - ☞ For more information on upgrading StartLine parameters, see [“Start parameters and the compatibility library” on page 382](#).
 - ☞ For more information about creating connection descriptions that capture the *dbclient* command information, see [“Capturing *dbclient* command information” on page 384](#).
- ◆ If the database server is started in some other way, such as by a batch file or using a Windows NT service, you must reconfigure this so that the newer version of the database server is started instead.
 - ☞ For information on creating Version 10 data sources, see [“Working with ODBC data sources” \[SQL Anywhere Database Administration Guide, page 56\]](#).

3. **Upgrade the database server** This step must be carried out at the server computer.
 - ◆ As with any software upgrade, back up your database before upgrading.

- ◆ Install SQL Anywhere on the server computer.
 - ◆ Start the Version 10 database server on the database.
4. **Use the new data source** This step must be carried out at each client computer. You need to use the new connection description to connect to the newer version of the database server. With this step, you are using all current software and have completed your upgrade. The database itself does not need to be upgraded to work with existing applications.

Upgrade notes for PowerBuilder and InfoMaker users

Users of Sybase PowerBuilder and InfoMaker should make some changes in order to obtain full functionality with SQL Anywhere Version 10.

The pbodb80.ini file

PowerBuilder and InfoMaker use a file named *pbodb80.ini* to hold ODBC data source information. The 80 in the file name may be different, depending on the version you have. For each ODBC driver it provides such things as DDL syntax, default DBParm options, valid function names and special data types.

If your *pbodb80.ini* file does not have a SQL Anywhere section, PowerBuilder and InfoMaker default to a core syntax. This limits the operations you can carry out using these tools. For example you cannot create, alter, or drop primary and foreign keys.

Upgrading your
pbodb80.ini file

To obtain complete functionality with PowerBuilder and InfoMaker, you need to upgrade your *pbodb80.ini* file.

❖ To upgrade your `pbodb80.ini` file

1. Make a backup copy of your existing `pbodb80.ini` file.
2. Add an SQL Anywhere section to the working copy of the file containing the same information as the existing Sybase SQL Anywhere section:

```
[SQL Anywhere]
PBSyntax='WATCOM50_SYNTAX'
PBDateTime='STANDARD_DATETIME'
PBFunctions='WATCOM_FUNCTIONS'
PBDefaultValues='autoincrement,current date,current
                time,current timestamp,timestamp,null,user'
PBDefaultCreate='YES'
PBDefaultAlter='YES'
PBDefaultExpressions='YES'
DelimitIdentifier='YES'
PBDateTimeInvalidInSearch='NO'
PBTimeInvalidInSearch='YES'
PBQualifierIsOwner='NO'
PBSpecialDataTypes='WATCOM_SPECIALDATATYPES'
IdentifierQuoteChar='"'
PBSystemOwner='sys, dbo,rs_systabgroup'
PBUseProcOwner='YES'
SQLSrvrTSName='YES'
SQLSrvrTSQuote='YES'
SQLSrvrTSDelimit='YES'
ForeignKeyDeleteRule='Disallow if Dependent Rows Exist
                      (RESTRICT),Delete any Dependent Rows (CASCADE),Set
                      Dependent Columns to NULL (SET NULL)'
TableListType='GLOBAL TEMPORARY'
```

Using the compatibility library

The compatibility library is a dynamic library (a DLL on PCs, shared library on UNIX) that enables embedded SQL applications to work with both Version 5 and Version 10 database servers. This section describes how the compatibility library works.

Who needs to read this section?

You should read this section if you are upgrading SQL Anywhere Version 5 to SQL Anywhere Version 10, and have existing embedded SQL applications that you need to work with the Version 10 server.

The Version 5 embedded SQL interface library

All client computers running SQL Anywhere Version 5 applications, whether connecting over a network or to a personal server, have a SQL Anywhere Version 5 interface library.

For Windows operating systems, this library is a DLL named *db150t.dll*.

How Version 5 client applications locate the interface library

Version 5 client applications locate the interface library in one of the following ways:

- ◆ **ODBC applications** ODBC applications connect to a SQL Anywhere database using the SQL Anywhere ODBC driver. The SQL Anywhere Version 5 ODBC driver calls functions in the Version 5 embedded SQL interface library.
- ◆ **Embedded SQL applications** Embedded SQL applications for Windows operating systems call into the interface library. In these calls, the interface library is referenced by name—*db150t.dll*.

The SQL Anywhere Version 5 ODBC driver is an embedded SQL application.

Using the compatibility library

The compatibility library is *optionally* installed as part of the Version 10 client software. It provides support for two interface libraries at the same time. You should check your installation to confirm that it is installed.

The setup program should ensure that your application calls the compatibility library instead of the Version 5 interface library.

The setup program carries out the following steps to ensure that applications call the interface library.

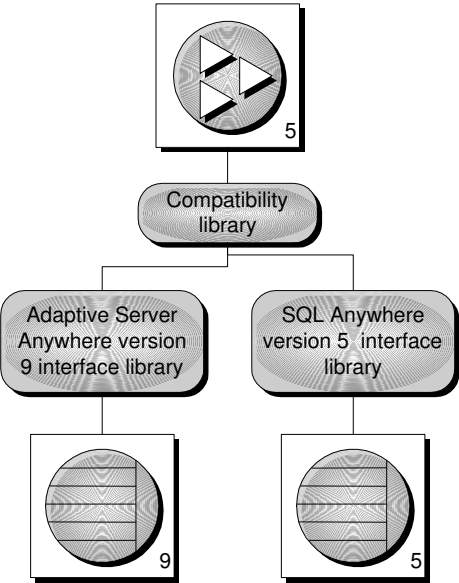
- ◆ The compatibility library has the same name as the SQL Anywhere Version 5 interface library. For example, on Windows NT, the compatibility library is named *dbl50t.dll*.
- ◆ The compatibility library is installed into the same directory as other Version 10 software.
- ◆ The Version 10 installation directory is placed ahead of the Version 5 directory in the system path. This ensures that applications locate the compatibility library ahead of the Version 5 interface library.
- ◆ A Version 5 interface library is installed into the same directory as the compatibility library, but with the name *dbl50to.dll*. When the compatibility library is accessing Version 5 servers, it calls this interface library.
- ◆ The Version 10 interface library is installed into the same directory as the compatibility library. It has the name *dblib10.dll* on Windows NT. When the compatibility library is accessing Version 10 servers, it calls this interface library.

If you have problems using the compatibility library, you should check the order of the directories in your path, and ensure that the Version 10 location is ahead of the Version 5 location in the path.

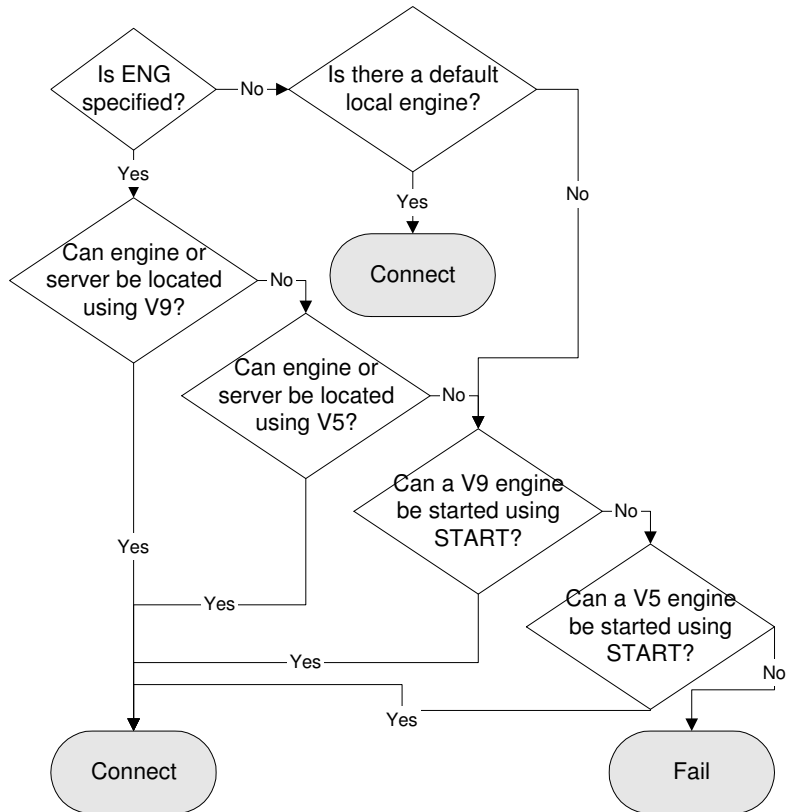
How the compatibility library works

Using the supplied connection string, the compatibility library attempts to connect to an SQL Anywhere database using the SQL Anywhere version 10 interface library. If this attempt fails, it attempts to connect to a SQL Anywhere database using the SQL Anywhere Version 5 library.

The following figure illustrates how the compatibility library enables communications to both a SQL Anywhere Version 5 and an SQL Anywhere Version 10 database server. The number in the lower right hand corner of the boxes indicates the version of the software component.



The following figure illustrates the algorithm used by the compatibility library to connect to a server:



File locations and the compatibility library

The compatibility library and the Version 5 interface library have the same file name (*db150.dll*). For your application to use the compatibility library, it must locate it ahead of the Version 5 interface library when it searches for DLLs.

To ensure that your application locates the compatibility library ahead of the Version 5 interface library you must understand how your application searches for DLLs.

Searching for DLLs

The Version 10 installation program does not necessarily place the Version 10 executable directory ahead of the Version 5 directory in the system path, so any application that uses the path to searching for *db150.dll* may have to be changed to find the compatibility library ahead of the Version 5 interface library.

The Version 5 ODBC driver is in the same directory as the Version 5

interface library, and so locates this library instead of the Version 10 library. To enable Version 5 ODBC applications to connect to Version 10 database servers, you can either rename the Version 5 interface library, or copy the Version 10 compatibility library along with *db150to.dll* into your Version 5 directory.

Testing to see which library is located

You can test to see which library is located in the following ways:

- ◆ Attempt to connect to a Version 10 database server. You cannot connect using the Version 5 interface library.
- ◆ Specify a database file parameter (DBF) and no start line in your connection string. If the version 5 interface library is located, the SQL Anywhere version 5 standalone database engine is started. If the compatibility library is located, the version 10 personal database server is started.

Start parameters and the compatibility library

Applications using a connection string that includes a StartLine connection parameter face some additional issues in upgrading.

The StartLine parameter provides explicit instructions for starting the database engine or the SQL Anywhere Client executable. Sample StartLine parameters are as follows:

- ◆ **Standalone application** A sample StartLine parameter for a Version 5 standalone application is as follows:

```
dbeng50.exe -c 8M
```

- ◆ **Network client application** A sample StartLine parameter for a Version 5 network client application is as follows:

```
dbclient.exe -x tcpip
```

In the current version of the software, the *dbeng50.exe* executable is replaced by the personal database server *dbeng10.exe*. The *dbclient.exe* executable is no longer required.

❖ To upgrade StartLine parameters

1. The procedure depends on where your connection parameters are stored.
 - ◆ If your connection parameters are stored outside the application itself, then you need to alter the connection parameters to use the appropriate *dbeng10.exe* executable name instead of *dbeng50.exe*. All the information on the *dbclient* command line can be rephrased in terms of other connection parameters. For information, see

“Connection Parameters and Network Protocol Options” [[SQL Anywhere Database Administration Guide](#), page 199].

- ◆ If your connection parameters are hard-wired into your application, you must relink your application with a new connection string. There are many possible configurations of client command lines and connection parameters. Be sure you test any solution thoroughly before deploying.

Capturing dbclient command information

Version 5 client applications that connect to a database server do so via the SQL Anywhere Client, an executable named *dbclient.exe*. The client executable command line contains information needed to locate a server, including the following:

- ◆ **Default server name** The server name on the client command line is the default server name. When a client executable is running, the application does not need to supply a server name in order to connect to the default server.
- ◆ **Network protocol options** A listing of network protocols to use together with a set of protocol options specifies where the client executable is to look as it attempts to locate a server.
- ◆ **Client/Server communication tuning** A set of parameters allows the packet size, buffer size, and so on to be tuned for optimum performance.

In Version 10, this information is held in an ODBC data source along with other connection information. As there is no longer a client executable, there is no longer a client command line. In Version 10, embedded SQL applications can use ODBC data sources as a source of connection parameters.

Client command line
scope

Only one Version 5 client executable can be run at a time, and it may be used by more than one application and handle connections to more than one server. The command information is therefore global to the computer.

How to capture client command information

During upgrade to Version 10, you must ensure that Version 5 *dbclient* command information is captured in such a way that the Version 10 ODBC or embedded SQL applications can use it. You can do this in one of the following ways:

- ◆ **Place the information in an ODBC data source** If the information can be placed in a data source, you can use it with Version 10 ODBC or embedded SQL applications.
- ◆ **Use the SQLCONNECT environment variable** The SQLCONNECT environment variable contains a connection string. It is searched early in the process of establishing which connection parameters to use. You may be able to use SQLCONNECT settings to override connection parameters.

The way to capture client command information depends on where the information is located.

Where command
information is located

Your existing command line information may be held in one of the following places.

- ◆ **ODBC data source** The ODBC data source contains a *START* parameter that can hold a client executable command.
- ◆ **As a connection string** Your application may obtain client information (for example from an initialization file), and supply it in a connection string as the *START* parameter.
- ◆ **A batch file** You may have a batch file that includes a client executable command line as part of your startup process.
- ◆ **Under an icon** You may have a client executable command line under an icon on your desktop.

How to capture the
information

- ◆ **From an ODBC data source** The ODBC data source upgrade captures the information in an ODBC data source *START* parameter.
- ◆ **From a batch file or under an icon** You can move the *dbclient.exe* parameters into your CommLinks connection parameter.
- ◆ **Hard-wired connection strings** Only if you have a hard-wired connection string in your application (that is, one that cannot be edited), you must alter the source of the application and recompile.

Upgrading databases

To use some the new features of SQL Anywhere, you need to rebuild your database file. See [“Upgrading SQL Anywhere” on page 334](#).

Upgrading version 5 SQL Remote installations

SQL Remote installations include a consolidated database and many remote databases, together with a Message Agent at each site.

At each site, the Message Agent handles the sending and receiving of messages. The messages take the form of SQL statements, and the database server handles the actual execution of those SQL statements.

The upgrade requirements for SQL Remote are as follows:

- ◆ **No need to upgrade databases** Although it is not essential to upgrade databases for Version 10, it is recommended for best performance that you upgrade the database file format by unloading and reloading your database. There is no need for all databases to be upgraded at the same time.

☞ For instructions on unloading and reloading the database, see “[Unloading and reloading a database participating in replication](#)” [*SQL Remote User’s Guide*, page 181]. For instructions on upgrading the database version, see “[Upgrading SQL Anywhere](#)” on page 334.

- ◆ **Software upgrades can be one site at a time** Version 5 Message Agents can exchange messages with Version 10 Message Agents as long as the compression database option is set to a value of -1 (minus one). There is no need to upgrade software throughout the installation simultaneously.
- ◆ **Message Agent and server can be upgraded separately** The Message Agent is an embedded SQL application. Therefore, the database server can be upgraded before the Message Agent as long as the compatibility library is used. It is, however, recommended that you upgrade your Message Agent at the same time as the database server for performance reasons.

The Message Agent cannot be upgraded before the database server, as a new client application cannot work with a Version 5 server.

Replication is based on the transaction log, and when a database is unloaded and reloaded, the old transaction log is no longer available. For this reason, good backup practices are especially important when participating in replication.

Example

One approach to upgrading is as follows:

1. Upgrade the consolidated database server and Message Agent. Set the compression database option to -1 so that all messages are compatible with the Version 5 software at remote sites.

2. Over time, upgrade remote database servers and Message Agents. You can set the compression database option to a value other than -1 to take advantage of compression and encoding on messages being sent to the consolidated database server.
3. When all remote database servers and Message Agents are upgraded, set the compression database option at the consolidated site to a value other than -1.

Index

Symbols

#hook_dict table			
dbmlsync enhancement in version 10.0.0	77		
SQL Remote enhancement in version 10.0.0	96		
\$best_adapter_mac			
new feature in 9.0.2	136		
\$best_adapter_name			
new feature in 9.0.2	136		
\$best_ip			
new feature in 9.0.2	136		
\$best_network_name			
new feature in 9.0.2	136		
\$request_id			
new feature in 9.0.2	136		
_IP_CHANGED_			
new feature in 9.0.2	136		
1252NOR collation			
new feature in version 10.0.0	15		
1252SPA			
new feature in version 9.0.2	129		
1252SWEFIN collation			
new feature in version 9.0.2	129		
1254TRKALT collation			
new feature in version 9.0.1	160		
1255HEB collation			
new feature in version 9.0.2	129		
1256ARA collation			
new feature in version 9.0.2	129		
64-bit software			
new feature in 9.0.0	180		
874THAIBIN			
new feature in version 9.0.2	129		
950TWN collation			
deprecated in version 9.0.2	146		
950ZHO_HK			
new feature in version 9.0.2	129		
950ZHO_TW collation			
new feature in version 9.0.2	129		
A			
a_backup_db structure			
backup_writefile member not supported in version 10.0.0		63	
a_compress_db structure			
unsupported in version 10.0.0		63	
a_db_collation structure			
unsupported in version 10.0.0		61	
a_db_info structure			
compressed member not supported in version 10.0.0		63	
new feature in 8.0.2		217	
wrtbufsize member not supported in version 10.0.0		63	
wrtnamebuffer member not supported in version 10.0.0		63	
a_stats_line structure			
unsupported in version 10.0.0		63	
a_writefile structure			
unsupported in version 10.0.0		63	
AccentSensitive property			
new feature in version 10.0.0		12	
access plans			
behavior change in version 8.0.0		267	
new feature in 9.0.0		193	
new features in version 8.0.0		246	
accessibility			
new features in version 8.0.0		245	
ActiveSync			
new MobiLink feature in version 8.0.0		256	
new UltraLite feature in version 8.0.0		261	
ActiveSync provider activity log			
new feature in 9.0.1		165	
Adaptive Server Anywhere			<i>see</i> SQL Anywhere
renamed SQL Anywhere in version 10.0.0		36	
upgrading to version 10		333	
Adaptive Server Anywhere behavior changes version 9.0.2		146	
Adaptive Server Anywhere Console utility [dbconsole]			
enhancement in version 9.0.1		159	
Adaptive Server Anywhere new features version 9.0.2		126	
Adaptive Server Enterprise			

MobiLink ODBC driver behavior change in version 10.0.0	89	ALTER PUBLICATION statement	
SQL Remote support dropped in version 10.0.096		UltraLite enhancement in version 10.0.0	102
Adaptive Server Enterprise compatibility behavior change in 10.0.0	64	ALTER PUBLICATION statement [MobiLink] [SQL Remote]	
addBatch method		syntax change in version 10.0.0	59
new feature in version 10.0.0	31	ALTER SERVER statement	
Admin mode in Sybase Central		behavior change in version 10.0.0	60
MobiLink enhanced feature in version 10.0.0	68	ALTER STATISTICS statement	
administration enhancements		new feature in version 10.0.0	27
version 7.0.0	296	ALTER SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]	
administration tools		syntax change in version 10.0.0	59
behavior change in version 10.0.0	120	ALTER SYNCHRONIZATION USER statement [MobiLink]	
ADO.NET		syntax change in version 10.0.0	59
UltraLite support in version 9.0.2	138	ALTER TABLE statement	
ADO.NET 2.0 support		enhancement in version 10.0.0	8, 27
new feature in version 10.0.0	31	enhancement in version 9.0.1	156
AGENT connection parameter		new features in version 6.0.3	313
behavior change in version 6.0.3	320	syntax change in version 10.0.0	59
deprecated feature in version 8.0.1	235	ALTER VIEW statement	
aggregate function outer references		enhancement in version 9.0.1	156
behavior change in version 8.0.0	266	ALTER WRITEFILE statement	
aggregate functions		unsupported in version 10.0.0	63
new features in 9.0.0	181	alternate server names	
aliases		new feature in version 10.0.0	3
behavior change in version 7.0.2	286	AlternateServerName property	
troubleshooting	286	new feature in version 10.0.0	12
allow_snapshot_isolation option		an_expand_db structure	
new feature in version 10.0.0	3	unsupported in version 10.0.0	63
ALTER DATABASE statement		ansi_blanks option	
behavior change in version 10.0.0	36, 41	behavior change in version 10.0.0	46
enhancement in version 10.0.0	28	new feature in version 10.0.0	8
new calibration feature in version 8.0.0	242	ansi_integer_overflow option	
new feature in 8.0.2	214	behavior change in version 10.0.0	46
syntax change in version 10.0.0	59	ansi_nulls option	
ALTER DBSPACE statement		enhancement in version 9.0.1	156
new feature in 8.0.2	217	ansi_substring option	
ALTER DOMAIN statement		new feature in version 10.0.0	8
new feature in version 9.0.2	127	ansi_update_constraints option	
ALTER EVENT statement		new feature in version 7.0.0	308
new feature in version 7.0.0	296	ansinull option	
syntax change in version 10.0.0	59	behavior change in version 6.0.1	332
ALTER INDEX statement		API migration wizard	
enhancement in version 10.0.0	29	UltraLite using	357
new feature in 9.0.0	183	APIs	
ALTER MATERIALIZED VIEW statement		UltraLite enhancement in version 10.0.0	105
new feature in version 10.0.0	27		

UltraLite upgrading	355	behavior change in version 10.0.0	40
AppForge CrossFire		enhancement in version 10.0.0	13
new feature in version 9.0.2	138	new feature in version 6.0.2	322
AppForge Crossfire		Auditing property	
UltraLite behavior change in version 9.0.2	151	new feature in version 10.0.0	13
AppForge MobileVB		authenticate_parameters	
UltraLite component new feature in 8.0.2	221	new feature in in version 9.0	195
AppInfo connection parameter		automated scripting	
enhancement in version 10.0.0	13	new UltraLite feature in 8.0.1	233
AppInfo property		automatic timestamp conflict tolerance	
new feature in version 7.0.0	302	new feature in 8.0.1	232
application profiling		AutoStart connection parameter	
new feature in version 10.0.0	4	new features in version 6.0.1	330
applications		AWE cache	
UltraLite upgrading	355	enhancement in version 10.0.0	9
ApproximateCPUTime property		new feature in 9.0.0	187
new feature in version 10.0.0	10		
Arabic		B	
new feature in version 9.0.2	129	-b option	
archive backups		MobiLink [dbmlsrv9] new feature in 9.0.0	195
enhancement in version 9.0.1	159	backlog option	
new NetWare feature in version 7.0.0	304	removed in version 10.0.0	82
asademo.db file		backlog protocol option	
renamed in version 10.0.0	121	new feature in 9.0.2	134
ASAJDBC		BACKUP authority	
renamed in version 10.0.0	60	new feature in version 10.0.0	18
ASANY environment variable		backup database wizard	
renamed in version 10.0.0	66	enhancement in version 9.0.1	162
ASANYSH environment variable		BACKUP statement	
renamed in version 10.0.0	66	behavior change in version 10.0.0	60
ASANYSH8 environment variable		enhanced support for variables in version 9.0.1	
new features in version 8.0.0	256	156	
ASAODBC		enhancement in version 10.0.0	14
renamed in version 10.0.0	60	enhancement in version 9.0.1	159
ASCII		new feature in 8.0.2	212
UltraLite enhancement in version 10.0.0	106	new feature in version 6.0.2	322
assume_distinct_servers option		new feature in version 9.0.2	131
deprecated in version 9.0.2	146	new features in 8.0.1	231
ASTART connection parameter		new features in version 6.0.3	313
new features in version 6.0.1	330	new features in version 7.0.0	297
ASTMP environment variable		new features in version 7.0.1	289
new feature in version 6.0.3	313	backup utility [dbbackup]	
asynchronous I/O		enhancement in version 10.0.0	14
new feature in version 9.0.1	162	enhancement in version 9.0.1	159
ATTACH TRACING statement		backups	
new feature in version 10.0.0	4, 27	new features in version 8.0.0	248
auditing		BASE64_DECODE function	

enhancement in version 9.0.1	164	certificates	
UltraLite behavior change in version 10.0.0	103	behavior change in version 9.0.2	147
cache warming		where to obtain	225
new feature in version 9.0.1	158	-ch option	
CacheAllocated property		database server behavior change in version 10.0.0	
new feature in version 10.0.0	10	44	
CacheFile property		changes in version	
new feature in version 10.0.0	10	10.0.0	1
CacheFileDirty property		6.0.1	332
new feature in version 10.0.0	10	6.0.2	322, 324
CacheFree property		6.0.3	316
new feature in version 10.0.0	10	7.0.0	305
CachePanics property		7.0.1	288, 289, 291
new feature in version 10.0.0	10	7.0.2	282, 286
CacheScavenges property		7.0.3	278, 279
new feature in version 10.0.0	10	8.0.0	240, 256, 260, 266
CacheScavengesVisited property		8.0.1	228, 235
new feature in version 10.0.0	10	8.0.2	212, 223
CacheSizingStatistics property		9.0.0	178, 201
new feature in version 10.0.0	11, 25	9.0.1	154, 172
callback functions		9.0.2	126, 146
new feature in version 9.0.2	130	char_oem_translation option	
Capabilities property		behavior change in version 6.0.1	332
new feature in version 8.0.0	251	character set conversion	
CarverHeapPages property		behavior change in version 8.0.0	269
new feature in version 10.0.0	10, 11	enhancement in version 10.0.0	5
cascading deletes		new feature in 8.0.2	213
UltraLite behavior change in version 9.0.2	150	character sets	
CASE expressions		behavior change in version 8.0.0	270
new UltraLite feature in 9.0.2	140	conversion	328
case sensitivity		UltraLite enhancement in version 10.0.0	99
new feature in version 9.0.0	188	character-length semantics	
CASE statement		new feature in version 10.0.0	31
new features in version 8.0.0	244	CharSet connection parameter	
CaseSensitive property		new features in version 6.0.3	314
new feature in version 7.0.3	278	CharSet property	
case-sensitivity		enhancement in version 10.0.0	6
UltraLite upgrade consideration for	349	checkpoint logs	
catalog		improved storage in version 8.0.0	242
behavior change in version 10.0.0	47	new features in version 8.0.0	242
-cc option		checkpoints	
new database server feature in version 9.0.1	158	enhancement in version 10.0.0	7
.cdb		Checksum property	
file extension not supported in version 10.0.0	62	new feature in version 9.0.1	157
Certicom		checksums	
new feature in 8.0.2	215	enhancement in version 10.0.0	14
no longer issues certificates after July, 2002	225	new feature in version 9.0.1	157

Chinese			
new feature in version 9.0.2	129		
-cl option			
database server behavior change in version 10.0.0			
44			
client message stores			
QAnywhere no longer uses transaction log in			
version 10.0.0	95		
client/server communications			
new features in version 8.0.0	252		
ClientPort property			
new feature in version 9.0.2	132		
client-port specification			
new UltraLite feature in version 7.0.1	293		
clustered hash group by			
new feature in version 9.0.1	157		
clustered indexes			
new feature in 8.0.2	212		
-cm option			
MobiLink [mlsrv10] new in version 10.0.0	71		
CodeWarrior			
new UltraLite feature in version 8.0.0	262		
supported versions	209		
version 5 unsupported in version 8.0.0	274		
version 8 supported in 8.0.1	233		
CodeWarrior stationery			
UltraLite C++ Component new feature in 9.0.1			
168			
Collation utility (dbcollat)			
unsupported in version 10.0.0	61		
collations			
behavior change in version 6.0.3	319		
deprecated in version 8.0.0	272		
new feature in version 10.0.0	15		
new feature in version 7.0.2	282		
new feature in version 9.0.1	160		
new feature in version 9.0.2	129		
new features in version 6.0.3	312		
new features in version 8.0.0	255		
UltraLite enhancement in version 10.0.0	99		
UltraLite upgrade recommendation for	349		
collect_statistics_on_dml_updates option			
new feature in version 10.0.0	8		
CollectStatistics property			
new feature in version 10.0.0	11, 25		
column compression			
new feature in version 10.0.0	8		
Comm Requests Received statistic			
new feature in version 10.0.0	12		
command history dialog			
new feature in 9.0.0	194		
command line utilities			
multiple versions	341		
UltraLite multiple versions	349		
UltraLite upgrading	349		
upgrading	341		
CommBufferSize connection parameter			
enhancement in version 10.0.0	13		
COMMENT statement			
behavior change in version 10.0.0	60		
behavior change in version 8.0.0	269		
enhancement in version 10.0.0	27, 28		
comments			
behavior change in version 8.0.0	269		
COMMIT statement			
UltraLite dynamic SQL new feature in 9.0.1	167		
CommLinks connection parameter			
behavior change in version 8.0.0	270		
common table expressions			
new feature in 9.0.0	180		
communication encryption			
new feature in version 9.0.2	132		
COMPARE function			
new feature in version 8.0.0	244		
compatibility			
client/server	334		
databases and database servers	334		
issues	334		
UltraLite software upgrades	349		
compatibility library			
installation	378		
introduction	363		
locating	381		
ODBC applications	381		
upgrades	378		
compress database wizard			
enhancement in version 9.0.1	162		
unsupported in version 10.0.0	63		
COMPRESS function			
new feature in version 10.0.0	26		
new feature in version 9.0.1	155		
compressed B-tree indexes			
new features in version 8.0.0	241		
compressed columns			

new feature in version 10.0.0	8	behavior change in version 10.0.0	40
compressed databases		enhancement in version 10.0.0	13
unsupported in version 10.0.0	62	UltraLite API controls for	349
CompressedBTrees property		connection profiles	
database property deprecated in version 10.0.0	67	enhancement in version 10.0.0	113
new feature in version 8.0.0	251	connection properties	
compression		behavior change in version 10.0.0	37
enhancement in version 10.0.0	26	connection strings	
new features in version 8.0.0	252	enhancements in version 10.0.0	13, 39
Compression property		connection timeout	
unsupported in version 10.0.0	64	new MobiLink feature in version 8.0.0	258
CompressionThreshold connection parameter		CONNECTION_EXTENDED_PROPERTY	
enhancement in version 10.0.0	13	function	
computed columns		new feature in version 10.0.0	25
new features in version 6.0.3	312	CONNECTION_PROPERTY function	
recalculated by LOAD TABLE	235	behavior change in version 9.0.2	148
com.sybase.asa.ultralite.jdbc package		enhancement in version 10.0.0	24
name change in version 8.0.0	275	ConnectionName connection parameter	
CON connection parameter		new ODBC feature in version 7.0.2	283
new ODBC feature in version 7.0.2	283	connections	
concurrent uploading		choosing parameter control over connection	
new MobiLink feature in version 8.0.0	258	object	349
conduit configuration		console utility [dbconsole]	
new UltraLite feature in 8.0.1	233	enhancement in version 10.0.0	115
confirmation_handler		enhancement in version 9.0.1	159
new in version 10.0.0	79	ConsoleLogFile property	
CONFLICT function		new feature in version 10.0.0	25
new feature in version 9.0.2	126	new feature in version 9.0.1	163
SQL Remote new feature in version 9.0.2	137	ConsoleLogMaxSize property	
conflict resolution triggers		new feature in version 10.0.0	11, 25
new feature in version 9.0.2	126	consolidated databases	
SQL Remote new feature in version 9.0.2	137	upgrading	343
conflicted_deletes		constraints	
new behavior in version 10.0.0	84	enhancements in version 10.0.0	6
conflicted_inserts		new features in 9.0.0	182
new behavior in version 10.0.0	84	UltraLite enhancement in version 10.0.0	102
conflicted_updates		CONTINUE statement	
new behavior in version 10.0.0	84	enhancement in version 10.0.0	29
conn_auditing option		conventions	
new feature in version 10.0.0	13	documentation	x
ConnAuditing property		CONVERT function	
new feature in version 10.0.0	13	new behavior in version 8.0.0	270
connect dialog		new features in version 6.0.3	315
behavior change in version 9.0.1	173	converting NULL constants	
connection IDs		behavior change in version 10.0.0	44
enhancement in version 9.0.1	163	CORR function	
connection parameters		new feature in version 9.0.1	154

cost models			
new feature in version 8.0.0	241	deprecated in version 8.0.0	273
COUNT_SET_BITS function		removed in version 10.0.0	88
new feature in version 10.0.0	24	Create Synchronization Model wizard	
countUploadRows method (JdbcConnection class)		MobiLink new feature in version 10.0.0	68
new UltraLite feature in 8.0.1	233	CREATE SYNCHRONIZATION SITE statement	
COVAR_POP function		deprecated in version 8.0.0	273
new feature in version 9.0.1	154	removed in version 10.0.0	88
COVAR_SAMP function		CREATE SYNCHRONIZATION SUBSCRIPTION	
new feature in version 9.0.1	154	statement	
CPU		new MobiLink feature in version 8.0.0	256
behavior change in version 9.0.0	201	scheduling change in 8.0.2	225
-cr option		CREATE SYNCHRONIZATION TEMPLATE	
new database server feature in version 9.0.1	158	statement	
create backup images wizard		deprecated in version 8.0.0	273
enhancement in version 9.0.1	162	removed in version 10.0.0	88
CREATE COMPRESSED DATABASE statement		CREATE SYNCHRONIZATION USER statement	
unsupported in version 10.0.0	63	new MobiLink feature in version 8.0.0	256
Create Custom Collation wizard		scheduling change in 8.0.2	225
unsupported in version 10.0.0	61	CREATE TABLE statement	
CREATE DATABASE statement		enhancement in version 10.0.0	8, 27
behavior change in version 10.0.0	41	UltraLite dynamic SQL new feature in 9.0.1	167
enhancement in version 10.0.0	19	create write file wizard	
enhancement in version 9.0.1	157	unsupported in version 10.0.0	63
enhancements in version 10.0.0	28, 30	CREATE WRITEFILE statement	
new feature in version 9.0.0	188	unsupported in version 10.0.0	63
CREATE ENCRYPTED FILE statement		creating	
enhancement in version 10.0.0	28	databases, behavior change in version 8.0.0	266
CREATE EVENT statement		creator ID	
behavior change in 8.0.2	223	UltraLite enhancement in version 10.0.0	101
new feature in version 7.0.0	296	CrossFire	<i>see</i> AppForge CrossFire
CREATE EXPANDED DATABASE statement		CS connection parameter	
unsupported in version 10.0.0	63	new features in version 6.0.3	314
CREATE FUNCTION statement		CSCONVERT function	
new feature in 8.0.1	231	new feature in 8.0.2	213
CREATE INDEX statement		-ct option	
UltraLite dynamic SQL new feature in 9.0.1	167	behavior change in version 8.0.0	255
CREATE LOCAL TEMPORARY TABLE statement		CUBE operation	
enhancement in version 10.0.0	30	new feature in version 9.0.1	154
CREATE MATERIALIZED VIEW statement		CUME_DIST function	
new feature in version 10.0.0	27	new feature in version 9.0.1	154
CREATE PUBLICATION statement		CURRENT UTC TIMESTAMP	
UltraLite enhancement in version 10.0.0	102	new feature in version 8.0.0	244
CREATE SERVER statement		CURRENT_TIMESTAMP	
behavior change in version 10.0.0	60	behavior change in version 9.0.1	172
CREATE SYNCHRONIZATION DEFINITION		CURRENT_USER	
statement		behavior change in version 9.0.1	172
		CurrentLineNumber property	

new feature in version 10.0.0	10	database mirroring	
CurrentProcedure property		new feature in version 10.0.0	3
new feature in version 10.0.0	10	database options	
cursors		behavior change in version 10.0.0	45
behavior change in version 8.0.0	267	database properties	
new features in version 8.0.0	247	behavior change in version 10.0.0	37
new UltraLite feature in version 8.0.0	264	UltraLite enhancement in version 10.0.0	99
UltraLite enhancement in version 10.0.0	102	UltraLite new feature in 9.0.1	170
custom collations		database servers	
unsupported in version 10.0.0	61	behavior change on NetWare in version 10.0.0	62
custom_header protocol option		enhancement in version 10.0.0	20
new MobiLink feature in 9.0.2	136	name restrictions removed in version 10.0.0	39
new UltraLite feature in 9.0.2	141	databases	
-cv option		compression not supported in version 10.0.0	62
new database server feature in version 9.0.1	158	upgrading	334, 386
-cw option		upgrading file format	336
new feature in version 8.0.0	249	upgrading version 5	363
D		versions supported in SQL Anywhere 10.0.0	36
-d option		write files not supported in version 10.0.0	62
database server support removed in version 10.0.0	62	DataWindow.NET	
MobiLink [mlsrv10] removed in version 10.0.082		new feature in version 10.0.0	120
data migration wizard		date_format option	
new feature in 8.0.1	229	behavior change in version 10.0.0	46
data source utility [dbdsn]		new behavior in version 8.0.0	270
behavior change in version 10.0.0	61	DATEDIFF function	
behavior change in version 9.0.2	147	behavior change in version 6.0.3	319
enhancements in version 10.0.0	18	DATEPART function	
new feature in version 9.0.2	130	behavior change in version 7.0.2	283
new features in version 7.0.2	284	dates	
data sources		enhancement in version 10.0.0	16
new feature in version 9.0.2	130	db_backup function	
database administration utilities		DB_BACKUP_WRITEFILE parameter not	
Version 5	371	supported in version 10.0.0	64
database encryption		enhancement in version 10.0.0	14
new feature in version 9.0.2	132	DB_BACKUP_WRITEFILE parameter	
new UltraLite feature in version 8.0.0	260	unsupported in version 10.0.0	64
database file format		DB_CALLBACK_FINISH callback parameter	
new features in version 8.0.0	242	behavior change in version 10.0.0	38
UltraLite upgrading	349	DB_CALLBACK_START callback parameter	
upgrading	336	behavior change in version 10.0.0	38
database file formats		DB_EXTENDED_PROPERTY function	
old formats deprecated in version 9.0.2	146	behavior change in version 9.0.2	148
database files		enhancement in version 10.0.0	24
encryption in version 8.0.0	243	db_get_property function	
new features in version 8.0.0	243	new feature in version 8.0.0	247
		db_locate_servers function	
		new feature in version 7.0.1	288

db_locate_servers_ex function		behavior change in version 9.0.2	147
enhancements in version 10.0.0	32	DBExpand function	
new feature in version 9.0.2	128	unsupported in version 10.0.0	63
DB_PROPERTY function		dbexpand utility	
behavior change in version 9.0.2	148	unsupported in version 10.0.0	63
enhancement in version 10.0.0	24	dbfhide utility	
db_register_a_callback function		new feature in 8.0.2	215
behavior change in version 10.0.0	38	new feature in version 9.0.2	132
db_string_ping_server function		-dbfile option	
new feature in version 7.0.0	303	QAnywhere option deprecated in 9.0.2	151
dbasdesk.dll		DBFileFragments property	
name changed to mlasdesk.dll in 10.0.0	90	new feature in 8.0.1	231
dbasdev.dll		new feature in version 8.0.0	251
name changed to mlasdev.dll in 10.0.0	90	dbhist utility	
dbasinst utility		enhancement in version 10.0.0	19
name changed to mlasinst in 10.0.0	89	dbinfo utility	
-dbauser option		enhancement in version 10.0.0	19
QAnywhere option deprecated in 9.0.2	151	new features in version 8.0.0	255
dbbackup utility		dbinit utility	
enhancement in version 10.0.0	14	behavior change in version 10.0.0	41
enhancement in version 9.0.1	159	behavior change in version 8.0.0	266
DBChangeWriteFile function		enhancement in version 10.0.0	19
unsupported in version 10.0.0	63	enhancement in version 9.0.1	157
dbclient		new feature in version 9.0.0	188
upgrading	384	dbisql utility	
dbclient utility		behavior change in version 10.0.0	116
unavailable in version 7.0.0	307	enhancement in version 10.0.0	115
DBCcollate function		new features in version 8.0.0	245
unsupported in version 10.0.0	61	dbisqlg.exe	
DBCompress function		new feature in 9.0.0	193
unsupported in version 10.0.0	63	db150t.dlll	
dbconsole utility		locating	381
enhancement in version 10.0.0	115	dblic utility	
enhancement in version 9.0.1	159	new features in version 8.0.0	254
DBCreateWriteFile function		dblocate utility	
unsupported in version 10.0.0	63	behavior change in version 10.0.0	43
DBD::ASAny		enhancement in version 9.0.1	159
new Perl driver in version 9.0.1	158	new feature in version 10.0.0	20
dbdsn utility		dblog utility	
behavior change in version 10.0.0	61	behavior change in version 9.0.2	147
behavior change in version 8.0.0	271	DBLTM service type	
behavior change in version 9.0.2	147	new feature in version 10.0.0	20
enhancement in version 10.0.0	18	dbltm utility	
new feature in version 9.0.2	130	enhancement in version 10.0.0	19, 20
-dbeng option		dbmlctr9.dll	
QAnywhere option deprecated in 9.0.2	151	removed with MobiLink Windows Performance	
dberase utility		Monitor support in version 10.0.0	90

dbmlmon utility		new features in version 8.0.0	252
name changed to mlmon in version 10.0.0	89	DBTools interface	
dbmlsrv9		libdbtool9.so deprecated in version 9.0.2	146
behavior changes in version 8.0.0	273	dbtran utility	
name changed to mlsrv10 in version 10.0.0	89	behavior change in version 9.0.2	147
new features in version 7.0.1	291	dbuleng9	
dbmlsrv.mle		UltraLite new feature in 9.0.1	168
name changed to mlsrv10.mle in 10.0.0	90	dbunload utility	
dbmlstop utility		behavior change in version 10.0.0	41, 43
behavior change in version 8.0.0	273	enhancement in version 10.0.0	21
behavior change in version 8.0.1	236	enhancement in version 9.0.1	159
name changed to mlstop in version 10.0.0	89	dbupgrad utility	
new MobiLink behavior in version 8.0.0	258	behavior change in version 10.0.0	36, 41
dbmlsync integration component		dbvalid utility	
new feature in 9.0.1	165	behavior change in version 9.0.1	173
dbmlsync utility		enhancement in version 9.0.1	157
new feature in version 7.0.0	305	enhancements in version 10.0.0	21
new features in version 7.0.1	290	new features in version 8.0.0	248
new MobiLink feature in version 8.0.0	257	dbwrite utility	
programming interface in version 7.0.1	291	unsupported in version 10.0.0	63
dbmluser utility		dbxtract utility	
behavior change in version 8.0.0	273	behavior change in version 9.0.2	149
name changed to mluser in version 10.0.0	89	new features in version 7.0.0	305
-dbname option		-dc option	
QAnywhere option deprecated in 9.0.2	151	dbmlsync new feature in 9.0.1	165
dbns10 utility		-dd option	
new feature in version 10.0.0	20	MobiLink [mlsrv10] removed in version 10.0.0	82
dbping utility		DDE protocol	
behavior change in version 10.0.0	42	unsupported in version 7.0.0	307
enhancement in version 10.0.0	20	deadlock reporting	
new features in version 8.0.0	252	new feature in version 9.0.2	129
dbremote		Deadlocks tab	
new feature in version 9.0.2	137	new feature in version 10.0.0	113
dbshrink utility		DEALLOCATE statement	
unsupported in version 10.0.0	63	new feature in 8.0.2	214
dbspaces		DEBUG connection parameter	
behavior change in version 9.0.2	148	deprecated feature in version 8.0.1	235
DBStatusWriteFile function		new features in version 7.0.1	289
unsupported in version 10.0.0	63	debug_messages option	
dbsupport utility		enhancement in version 9.0.1	156
new feature in version 10.0.0	21	debugger	
dbsvc utility		new feature in version 6.0.3	312
behavior change in version 10.0.0	43	DebuggingInformation property	
behavior change in version 9.0.2	147	new feature in version 10.0.0	11, 25
enhancement in version 10.0.0	20, 33	DECL_FIXCHAR macro	
new feature in version 7.0.1	288	behavior change in version 7.0.0	309
new feature in version 9.0.2	129	DECL_LONGBINARY macro	

new UltraLite feature in version 8.0.0	264	write files not supported in version 10.0.0	62
DECL_LONGVARCHAR macro		deployment wizard	
new UltraLite feature in version 8.0.0	264	new feature in version 10.0.0	33
DECLARE CURSOR statement		deprecated features	
new features in version 7.0.0	302	version 10.0.0	1
DECLARE LOCAL TEMPORARY TABLE		version 8.0.1	235, 237
statement		version 8.0.2	224
behavior change in version 9.0.1	173	version 9.0.0	205
DECOMPRESS function		version 9.0.1	172
new feature in version 10.0.0	26	version 9.0.2	146
new feature in version 9.0.1	155	derived tables	
DECRYPT function		key join behavior change in version 10.0.0	59
new feature in version 9.0.1	155	UltraLite new feature in 9.0.1	168
dedicated_task option		DESCRIBE statement	
new feature in version 9.0.1	160	enhancement in version 10.0.0	115
default .SQL file editor		describe_java_format option	
new feature in 9.0.0	194	unsupported in version 10.0.0	40
DEFAULT UTC TIMESTAMP		Design mode in Sybase Central	
new feature in version 8.0.0	244	MobiLink new feature in version 10.0.0	68
default values		destination aliases	
behavior change in version 6.0.3	319	QAnywhere new feature in version 10.0.0	94
default_dbspace option		DETACH TRACING statement	
new feature in version 10.0.0	8	new feature in version 10.0.0	4, 27
default_isql_encoding option		devices	
new feature in version 9.0.1	160	UltraLite enhancement in version 10.0.0	98
DefaultCollation property		-dh option	
new feature in version 6.0.3	314	new feature in version 10.0.0	9
DefaultNCharCollation property		diagnostic tracing	
new feature in version 10.0.0	11	new feature in version 10.0.0	4
defragmenting		directory access servers	
new feature in version 8.0.0	248	new feature in 10.0.0	14
new UltraLite feature in version 8.0.0	262	disk full	
DELETE statement		new feature in 9.0.2	130
enhancement in version 10.0.0	29	disk space	
delete_old_logs option		new feature in version 9.0.2	130
new features in version 7.0.2	282	DISTINCT clause	
DeleteAllRows function		UltraLite enhancement in version 10.0.0	102
new UltraLite feature in version 8.0.0	264	distributed transactions	
deleting database files		new feature in version 7.0.0	297
new UltraLite feature in 8.0.2	221	DLL protocol option	
demo.db		behavior change in 10.0.0	66
new sample database in version 10.0.0	121	documentation	
DENSE_RANK function		conventions	x
new feature in version 9.0.1	154	new features in version 8.0.0	254
deploying		SQL Anywhere	viii
new UltraLite features in version 8.0.0	263	documentation enhancements	
registry entry for event viewer messages	132	version 10.0.0	119

version 9.0.2	144	new feature in version 7.0.0	299
domains		new feature in version 7.0.2	282
behavior change in version 10.0.0	42	dynamic SQL	
DOS		UltraLite new feature in version 10.0.0	98
unsupported UltraLite platform in version 8.0.0			
274			
DOUBLE data type		E	
new MobiLink feature in version 6.0.3	317	-e option	
new UltraLite feature in version 6.0.3	317	deprecated in version 8.0.0	272
download acknowledgement		QAnywhere option deprecated in 9.0.2	151
MobiLink default change in version 9.0	205	-ec option	
new MobiLink feature in version 8.0.0	257	new feature in 8.0.2	215
download stream		server option behavior change in version 9.0.2	147
new MobiLink behavior in version 8.0.0	257	-ec server option	
download_delete_cursor		behavior change in version 10.0.0	47
new behavior in 8.0.2	226	ECC	
DownloadOnly dbmsync extended option		Beta: ECC strong encryption not supported	123
MobiLink new feature in 9.0.0	196	ECC encryption	
download-only publications		UltraLite new feature in version 10.0.0	104
new in version 10.0.0	76	-ek option	
download-only synchronization		QAnywhere option deprecated in 9.0.2	151
behavior change in version 8.0.0	275	embedded SQL	
new UltraLite feature in version 8.0.0	261	deprecated in UltraLite 9.0.2	150
Driver connection parameter		UltraLite new feature in version 10.0.0	98
new feature in version 9.0.2	130	eMbedded Visual Basic	
DriveType property		UltraLite component new feature in 8.0.2	221
new feature in version 9.0.1	163	eMbedded Visual C++	
DROP INDEX statement		new UltraLite feature in version 8.0.0	262
UltraLite dynamic SQL new feature in 9.0.1	167	empty strings	
DROP PUBLICATION statement		behavior change in version 8.0.0	269
UltraLite enhancement in version 10.0.0	102	emulators	
DROP statement		QAnywhere support in 9.0.2	142
enhancement in version 10.0.0	27	ENC connection parameter	
DROP TABLE statement		new feature in 8.0.2	215
UltraLite dynamic SQL new feature in 9.0.1	167	ENCRYPT function	
ds dbmsync extended option		new feature in version 9.0.1	155
MobiLink new feature in 9.0.0	196	encryption	
-ds option		behavior change in version 9.0.2	147
MobiLink [dbmsync] new feature in 9.0.0	196	new feature in 8.0.2	215
-dsd option		new features in version 8.0.0	243
MobiLink new feature in version 10.0.0	71	new UltraLite feature in version 8.0.0	260
DSN connection parameter		ENCRYPTION connection parameter	
behavior change in version 10.0.0	44	behavior change in version 10.0.0	47
-dt option		Encryption connection parameter	
MobiLink new feature in version 10.0.0	71	new feature in 8.0.2	215
new feature in version 10.0.0	10	encryption key changes	
dynamic cache sizing		new feature in version 10.0.0	28
		encryption keys	

restrictions removed in version 10.0.0	39	enhancement in version 10.0.0	30
Encryption property		new features in version 7.0.0	304
behavior change in version 10.0.0	39	EVENT_PARAMETER function	
EncryptionScope property		behavior change in 9.0.2	148
new feature in version 10.0.0	12	new feature in version 7.0.2	284
end_publication		EXCEPT operation	
new feature in version 9.0.0	195	new feature in 9.0.0	180
entity-relationship tab		EXECUTE IMMEDIATE statement	
new feature in version 10.0.0	113	behavior change in version 9.0.1	174
environment variables		enhancement in 9.0.0	182
behavior change in version 10.0.0	42	exit codes	
enhancement in version 10.0.0	15	Interactive SQL new feature in version 7.0.2	282
erase utility [dberase]		MobiLink [dbmlsync] enhancement in version 9.0.0	196
behavior change in version 9.0.2	147	EXIT statement	
error codes		behavior change in version 10.0.0	117
UltraLite behavior change in 9.0.0	209	enhancement in version 9.0.0	183
error context reporting		new feature in 8.0.2	214
new feature in 8.0.1	232	expanded mode	
error handling		new UltraLite feature in 9.0.2	139
new MobiLink feature in version 8.0.0	259	EXPLANATION function	
UltraLite C/C++ enhancement in 9.0.1	168	new feature in 8.0.2	213
error logs		export wizard	
SQL Remote new feature in 8.0.2	221	enhancement in version 9.0.1	161
error messages		EXPRTYPE function	
new features in version 8.0.0	245	new feature in 9.0.0	183
error reporting		extended_join_syntax option	
new feature in version 10.0.0	120	new feature in version 6.0.2	323
new MobiLink feature in version 8.0.0	259	external functions	
new UltraLite feature in version 8.0.0	262	new features in version 7.0.0	301
error strings		external stored procedures	
UltraLite enhancement in version 10.0.0	101	new features in version 7.0.0	301
error_handler		extract database wizard	
new in version 10.0.0	79	behavior change in version 10.0.0	45
ERRORMSG function		UltraLite using	353
new feature in version 8.0.0	245	extraction utility	
ESTIMATE_SOURCE function		behavior change in version 7.0.0	309
new behavior in version 8.0.0	271	behavior change in version 9.0.2	149
-esu option		new features in version 7.0.0	305
MobiLink new feature in version 10.0.0	71		
euro		F	
new features in version 6.0.3	312	failed rebuilds	
event handling		upgrading databases to version 10	339
new feature in version 7.0.0	296	fast launcher	
event hooks		new feature in 9.0.0	193
MobiLink new behavior in 9.0.1	175	fatal errors	
new features in version 7.0.1	290	Unix enhancement in version 10.0.0	33
event log			

-fd option		new features in version 7.0.0	298
QAnywhere new feature in version 10.0.0	93	FIRST clause	
Federal Rehabilitation Act		new features in version 7.0.0	301
section 508	245	first day of week	
feedback		behavior change in version 7.0.2	283
documentation	xiv	first row optimization option	
providing	xiv	new feature in version 8.0.0	242
fetch operation		first_day_of_week option	
new feature in 8.0.2	217	new feature in version 7.0.2	283
file formats		FirstOption property	
new features in version 8.0.0	242	new feature in version 10.0.0	11
UltraLite new feature in version 10.0.0	97	font selection	
UltraLite upgrading	349	new feature in version 10.0.0	114
upgrading	336	FOR XML clause	
file hiding utility [dbfhide]		new feature in 9.0.0	178
new feature in 8.0.2	215	for_xml_null_treatment option	
new feature in version 9.0.2	132	new feature in 9.0.0	178
file names		foreign key constraints	
.cdb extension not supported in version 10.0.0	62	behavior change in version 10.0.0	38
.wrt extension not supported in version 10.0.0	62	foreign key property sheet	
file-based downloads		enhancement in version 9.0.1	162
MobiLink new feature in 9.0.0	194	-fr option	
FileDataSourceName connection parameter		MobiLink [dbmlsrv9] new feature	208
new UNIX features in version 7.0.3	278	QAnywhere new feature in version 10.0.0	93
FILEDSN connection parameter		fragmentation	
new UNIX features in version 7.0.3	278	new features in version 8.0.0	249
files		FreePageBitMaps property	
new feature in version 7.0.0	301	database property deprecated in version 10.0.0	67
FileSize property		new feature in version 8.0.0	251
writefile dbspace support removed in version		FreePages property	
10.0.0	64	writefile dbspace support removed in version	
Finnish		10.0.0	64
new feature in version 9.0.2	129	FROM clause	
FIPS		enhancement in version 9.0.1	158
behavior change in version 10.0.0	39	new feature in 8.0.2	215
Beta: FIPS strong encryption not supported	123	setting isolation levels	229
enhancement in version 10.0.0	17	FTP message type	
UltraLite new synchronization feature in version		new NetWare feature in version 7.0.1	291
10.0.0	99	new SQL Remote feature in version 6.0.3	316
FIPS option		-ftr option	
MobiLink [mlsrv10] new feature in version		MobiLink [dbmlsrv10] new feature in version	
10.0.0	77	10.0.0	72
new feature in version 9.0.2	132	full outer joins	
FipsMode property		new feature in version 8.0.0	244
new feature in version 9.0.2	132	functions	
firewalls		new feature in 8.0.2	213
new features in version 6.0.3	313		

G

gencert utility	
enhanced in version 8.0.2	220
GET_BIT function	
new feature in version 10.0.0	24
GET_IDENTITY	
new feature in 8.0.1	229
GetColumnSQLType method	
new UltraLite feature in 8.0.1	233
getLastDownloadTime method (JdbcConnection class)	
new UltraLite feature in 8.0.1	233
getLastIdentity method (JdbcConnection class)	
new UltraLite feature in 8.0.1	233
getPropertyNames	
QAnywhere C++ function removed in version 10.0.0	95
global autoincrement	
behavior change in version 8.0.1	236
new feature in version 7.0.0	297, 305
new UltraLite feature in version 8.0.0	261
global temporary tables	
enhancements in version 10.0.0	30
new feature in 8.0.2	214
global variables	
new feature in 8.0.2	217
global_database_id option	
behavior change in version 8.0.1	236
GNU GCC tools	
new UltraLite feature in version 7.0.1	292
graphical plan display	
new feature in version 8.0.0	246
new features in 9.0.0	193
graphical plan with statistics	
new feature in 8.0.2	213
graphical plans	
and work tables	213
behavior changes in version 10.0.0	116
new feature in 8.0.2	213
GRAPHICAL_PLAN function	
new feature in 8.0.2	213
GROUPING SETS operation	
new feature in version 9.0.1	154
-gs option	
behavior change in version 6.0.1	332
-gtc option	
new in version 10.0.0	10

GUIDs

new feature in 8.0.2	212
gzip algorithm	
new feature in version 10.0.0	26

H

handle_odbc_error	
new feature in version 7.0.1	291
handling events	
new feature in version 7.0.0	296
hash antisemijoin	
new feature in 8.0.2	213
HASH function	
enhancement in version 10.0.0	26
new feature in version 9.0.1	155
hash joins	
new feature in version 8.0.0	241
hash semijoin	
new feature in 8.0.2	213
hash size	
deprecated in version 8.0.0	272
new feature in version 7.0.0	299
Heaps Carver statistic	
new feature in version 10.0.0	12
Heaps Query Processing statistic	
new feature in version 10.0.0	12
Heaps Relocatable Locked statistic	
new feature in version 10.0.0	12
Heaps Relocatable statistic	
new feature in version 10.0.0	12
HeapsCarver property	
new feature in version 10.0.0	10, 11
HeapsLocked property	
new feature in version 10.0.0	10, 11
HeapsQuery property	
new feature in version 10.0.0	10, 11
HeapsRelocatable property	
new feature in version 10.0.0	11
Hebrew	
new feature in version 9.0.2	129
hibernation	
new feature in version 8.0.0	254
histogram utility [dbhist]	
enhancement in version 10.0.0	19
histograms	
behavior change in version 9.0.2	146
Histograms property	

database property deprecated in version 10.0.0	67	QAnywhere transmission rule variable removed	
new feature in version 8.0.0	251	in version 10.0.0	95
host protocol option		ias_CurrentDayOfWeek	
behavior change in version 10.0.0	43	QAnywhere transmission rule variable removed	
HotSync conduit		in version 10.0.0	95
troubleshooting assistance in version 9.0.0	197	ias_CurrentMonth	
HotSync conduit configuration		QAnywhere transmission rule variable removed	
new UltraLite feature in 8.0.1	233	in version 10.0.0	95
HotSync progress		ias_CurrentYear	
new UltraLite feature in 8.0.1	233	QAnywhere transmission rule variable removed	
hover mode		in version 10.0.0	95
new SQL Remote feature in version 6.0.2	324	ias_MaxUploadSize	
HoverRescanThreshold dbmsync extended option		QAnywhere property new in version 10.0.0	93
new feature in version 9.0.0	196	IBM DB2	
hrt dbmsync extended option		MobiLink ODBC driver behavior change in	
new feature in version 9.0.0	196	version 10.0.0	89
HTTP		setup script name change in version 9.0.1	165
MobiLink security deprecated in 9.0.2	148	IBM DB2 8.2 CLI driver	
HTTP clients		MobiLink new support in version 10.0.0	89
new feature in version 9.0.2	128	icons	
HTTP server		used in manuals	xii
new feature in version 9.0.2	128	ICU	
HTTP synchronization		new feature in version 10.0.0	5
new UltraLite feature in version 7.0.1	293	-id option	
http_password protocol option		unsupported for dbtran	273
new MobiLink feature in 9.0.2	136	identifiers	
http_proxy_password protocol option		behavior change in version 10.0.0	39
new MobiLink feature in 9.0.2	136	behavior change in version 9.0.1	174
http_proxy_userid protocol option		maximum length change in version 7.0.0	308
new MobiLink feature in 9.0.2	136	square brackets can delimit in 8.0.1	229
http_userid protocol option		Idle connection parameter	
new MobiLink feature in 9.0.2	136	new feature in version 8.0.0	253
HTTPS		IdleTimeout property	
MobiLink new feature in 8.0.2	220	new feature in 8.0.1	231
HTTPS synchronization		new feature in version 10.0.0	25
new UltraLite features in 8.0.2	222	IF expressions	
HttpServiceName property		new UltraLite feature in 9.0.2	140
new feature in version 10.0.0	10	ignore parameter	
		new feature in 9.0.2	134
		ignored_deletes	
		new behavior in version 10.0.0	84
		ignored_inserts	
		new behavior in version 10.0.0	84
		ignored_updates	
		new behavior in version 10.0.0	84
		image backups	
		enhancement in version 9.0.1	159
I			
iAnywhere JDBC driver			
new feature in 8.0.2	216		
iAnywhere.Data.UltraLite namespace			
ADO.NET interface in version 9.0.2	138		
iAnywhere.UltraLite namespace			
UltraLite removed in version 10.0.0	110		
ias_CurrentDayOfMonth			

IMAGE data type			
new MobiLink feature in version 7.0.0	305	new feature in version 9.0.2	131
storage enhancement in version 7.0.0	300	intent locks	
import wizard		new feature in version 10.0.0	7
enhancement in version 9.0.1	161	Interactive SQL	
importing		batch operation new features in version 8.0.0	247
new features in version 7.0.0	301	behavior change in version 10.0.0	116
Index Consultant		behavior changes in version 10.0.0	116
enhancement in version 10.0.0	15	enhancement in version 10.0.0	115
index hints		enhancements in version 10.0.0	114
new feature in version 9.0.1	158	enhancements in version 9.0.1	161
index sharing		new feature in 8.0.2	214
new feature in version 10.0.0	35	new features in version 7.0.2	282
indexes		new features in version 8.0.0	245
behavior change in version 10.0.0	38	new UltraLite feature in 9.0.2	141
enhancements in version 10.0.0	6, 35	UltraLite new feature in version 10.0.0	100
new feature in 8.0.2	212	Version 5	371
new features in version 7.0.0	299	Interactive SQL behavior changes	
new features in version 8.0.0	241	version 10.0.0	116
indexes on functions		Interactive SQL new features	
new feature in 9.0.0	181	version 10.0.0	113
InfoMaker		Interactive SQL utility [dbisql]	
using with SQL Anywhere	376	enhancement in version 10.0.0	115
information utility [dbinfo]		interfaces	
enhancement in version 10.0.0	19	UltraLite new feature in version 10.0.0	100
INI files		Interop.UltraLiteAFLib.dll	
new feature in version 9.0.2	132	UltraLite behavior change in version 9.0.2	151
initialization utility [dbinit]		INTERSECT operation	
behavior change in version 10.0.0	41	new feature in 9.0.0	180
enhancement in version 10.0.0	19	intra-query parallelism	
enhancement in version 9.0.1	157	new feature in version 10.0.0	3
new feature in version 9.0.0	188	invalid_extensions option	
initialize database utility		new feature in version 10.0.0	96
upgrading databases with	354	IOParallelism	
Input Method Editors		new feature in version 10.0.0	12
behavior change in version 9.0.1	173	IPM_Receive control parameter	
INPUT statement		behavior change in version 9.0.2	149
enhancement in version 9.0.1	160	IPv6 support	
INSERT statement		new feature in version 10.0.0	16
enhancement in version 10.0.0	29	UltraLite new feature in version 10.0.0	104
new feature in 8.0.2	212, 214	IPX	
InstallShield projects		deprecated in version 7.0.0	307
new feature in 9.0.1	158	IPX protocol	
InstallShield templates		unsupported in version 8.0.0	271
new feature in version 7.0.1	289	ISDATE function	
integrated logins		new feature in version 8.0.0	245
enhancement in version 9.0.1	159	IsEccAvailable property	
		new feature in version 10.0.0	11

IsFipsAvailable property		unsupported in version 10.0.0	40
new feature in version 9.0.2	132	java_input_output option	
IsJavaAvailable property		unsupported in version 10.0.0	40
new feature in version 8.0.0	251	java_input_output property	
unsupported in version 10.0.0	41	unsupported in version 10.0.0	41
IsNetworkServer property		java_location option	
new feature in version 6.0.3	314	new feature in version 10.0.0	41
ISNUMERIC function		java_main_userid option	
new feature in version 8.0.0	245	new feature in version 10.0.0	41
isolation levels		java_namespace_size option	
setting in the FROM clause	229	unsupported in version 10.0.0	40
isolation_level option		java_page_buffer_size option	
enhancement in version 10.0.0	3	unsupported in version 10.0.0	40
isql_maximum_displayed_rows option		JavaGlobFix property	
new feature in version 10.0.0	115	unsupported in version 10.0.0	41
isql_plan option		JavaHeapSize property	
behavior change in version 10.0.0	116	unsupported in version 10.0.0	41
behavior change in version 8.0.0	272	java.io	
isql_show_multiple_result_sets option		new feature in version 6.0.3	315
new feature in version 10.0.0	115	java.lang.thread package	
IsRsaAvailable property		new feature in version 6.0.3	315
new feature in version 10.0.0	11	java.math.BigInteger class	
IsRuntimeServer property		new feature in version 7.0.0	304
new feature in version 7.0.0	302	java.net.PlainDatagramSocketImpl class	
ISYSFKEY system table		new feature in version 7.0.1	289
new feature in version 10.0.0	35	JavaNSSize property	
ISYSIDXCOL system table		unsupported in version 10.0.0	41
new feature in version 10.0.0	35	jConnect	
ISYSPHYSIDX system table		enhancement in version 10.0.0	36
new feature in version 10.0.0	35	new feature in version 7.0.0	298
		new features in version 6.0.1	330
		using to connect to Interactive SQL not supported	
		in version 10.0.0	117
		using to connect to SQL Anywhere Console	
		(dbconsole) not supported in version 10.0.0	117
		using to connect to Sybase Central not supported	
		in version 10.0.0	117
		version 4.5 support removed in version 10.0.0	61
		-jconnect option	
		not supported for dbconsole in version 10.0.0	117
		not supported for Interactive SQL in version	
		10.0.0	117
		JDBC	
		new feature in version 8.0.0	253
		new features in version 6.0.3	315
		JDBC escape syntax	
		new features in version 8.0.0	246

J

Java

behavior change in version 10.0.0	40
new features in version 6.0.1	329
security manager new feature in version 8.0.0	254

Java 2

new feature in version 8.0.0	253
------------------------------	-----

Java in the database

behavior change in version 10.0.0	40
behavior change in version 8.0.0	266
behavior change in version 9.0.0	201

Java synchronization logic

new MobiLink feature in version 8.0.0	256
---------------------------------------	-----

Java VM property

new feature in version 10.0.0	41
-------------------------------	----

java_heap_size option

JDBC JAR file			
behavior change in version 8.0.0	268	new feature in version 10.0.0	11
JDBC ResultSet methods		LastOption property	
new UltraLite feature in 8.0.1	233	new feature in version 10.0.0	11
JDK		LastPlanText property	
new features in version 8.0.0	253	new feature in version 10.0.0	10
UltraLite required version	208	LastServerProperty property	
JDKVersion property		new feature in version 10.0.0	11
unsupported in version 10.0.0	41	LastStatement property	
join algorithms		new feature in version 8.0.0	250
new feature in version 8.0.0	241	lateral derived tables	
joins		new feature in 9.0.0	182
new feature in 8.0.2	213	lazy checkpoints	
K		new feature in version 10.0.0	7
keep_alive		LazyClose connection parameter	
option removed in MobiLink version 9.0.0	208	new feature in 8.0.2	215
Kerberos		LCLOSE connection parameter	
new feature in version 10.0.0	18	new feature in 8.0.2	215
key joins		LDAP authentication	
behavior change in version 10.0.0	59	enhancement in version 10.0.0	16
keyset-driven cursors		MobiLink enhancement in 9.0.1	164
new feature in version 8.0.0	247	libdbtool9.so	
Kyocera 7135		deprecated in version 9.0.2	146
support added in version 10.0.0	80	license utility [dblic]	
L		new feature in version 7.0.2	282
-la_port option		licensing	
QAnywhere option deprecated in version 10.0.0	95	new feature in version 7.0.2	282
language selection utility [dblang]		new features in version 8.0.0	254
deprecated feature in version 9.0.1	172	limitations	
new feature in version 7.0.2	283	beta	123
LargeProcedureIDs property		LINKS connection parameter	
database property deprecated in version 10.0.0	67	behavior change in version 8.0.0	270
new feature in version 8.0.0	251	Linux	
last download time		new feature in version 9.0.1	162
new UltraLite feature in version 8.0.0	264	new features in version 10.0.0	33
last download timestamp		LIST function	
always passed first	206	enhancement in 9.0.0	181
new MobiLink feature in version 8.0.0	260	new features in version 7.0.0	303
new UltraLite feature in 8.0.2	222	ordering	267
LAST_BACKUP operation		Listeners	
new feature in version 10.0.0	14	MobiLink new feature in 9.0.0	194
LastConnectionProperty property		liveness	
new feature in version 10.0.0	11	new MobiLink features in version 8.0.0	260
LastDatabaseProperty property		liveness_timeout option	
		MobiLink client protocol option replaced in	
		version 10.0./0	75
		removed in version 10.0.0	82
		LivenessTimeout property	

new feature in 8.0.1	231	new feature in 8.0.1	231
LOAD TABLE statement		new feature in version 8.0.0	251
behavior change in version 10.0.0	61	logging	
behavior change in version 7.0.3	279	enhancement in version 10.0.0	30
enhanced support for variables in version 9.0.1		new MobiLink behavior in version 8.0.0	259
156		logical indexes	
enhancement in version 9.0.1	155	new feature in version 10.0.0	35
enhancements in version 10.0.0	28	login as a service privilege	
new feature in 8.0.2	214	granted by dbsvc utility	43
new features in version 6.0.3	313	login_mode option	
load XML to database utility		behavior change in version 10.0.0	46
upgrading databases with	352	login_procedure option	
LOCAL protocol option		behavior change in version 8.0.0	271
new feature in 8.0.1	231	new features in version 7.0.3	278
local temporary tables		new features in version 8.0.0	251
new feature in 8.0.2	214	LoginTime property	
LocalOnly protocol option		new feature in version 9.0.2	132
new feature in 8.0.1	231	LogMaxSize connection parameter	
LOCK TABLE statement		enhancement in version 10.0.0	13
new feature in version 7.0.0	301	LONG BINARY data type	
LockCount property		new MobiLink feature in version 7.0.0	305
new feature in version 10.0.0	10, 12	storage enhancement in version 7.0.0	300
LockedCursorPages property		LONG VARBIT data type	
new feature in version 10.0.0	10	new feature in version 10.0.0	31
locking		LONG VARCHAR data type	
enhancements in version 10.0.0	7	new MobiLink feature in version 7.0.0	305
LOG connection parameter		storage enhancement in version 7.0.0	300
new features in version 7.0.1	289	-lp option	
log files		QAnywhere new feature in version 10.0.0	95
enhancement in version 10.0.0	10	LTM	
log MobiLink issued ODBC statements		enhancement in version 10.0.0	19
new feature in 8.0.1	232	new features in version 8.0.0	255
log transfer manager utility [dbltm]		LTMGeneration property	
enhancement in version 10.0.0	19, 20	new feature in version 7.0.1	289
new features in version 6.0.3	316	LTMTrunc property	
log translation utility [dbtran]		new feature in version 7.0.0	302
behavior change in version 9.0.2	147		
new feature in version 6.0.2	323	M	
new features in version 6.0.3	316	MachineName property	
new SQL Remote feature in version 6.0.3	317	new feature in version 8.0.0	251
log_deadlocks option		maintenance plans	
new feature in version 9.0.2	129, 132	new feature in version 10.0.0	114
LOG_LEVEL		materialized views	
Redirector new feature in version 9.0	197	new feature in version 10.0.0	5
LogFile connection parameter		materialized_view_optimization option	
new features in version 7.0.1	289	new feature in version 10.0.0	9
LogFileFragments property		max_hash_size option	

deprecated in version 8.0.0	272	new feature in version 7.0.2	284
unsupported in version 10.0.0	62	min_password_length option	
max_plans_cached option		new feature in version 6.0.2	322
new feature in version 8.0.0	243	min_table_size_for_histogram option	
max_plans_cached property		behavior change in version 9.0.1	174
new feature in version 8.0.0	250	deprecated in version 9.0.2	146
max_temp_space option		new feature in version 8.0.0	243
new feature in version 10.0.0	8	MirrorLogDirectory dbmlsync extended option	
max_work_table_hash_size option		new feature in 9.0.2	136
deprecated in version 8.0.0	272	mixing C++ interfaces	
unsupported in version 10.0.0	62	UltraLite new feature in 9.0.1	168
MaxConcurrentReq property	11	ml_column	
MaxConnections property	11	new feature in version 10.0.0	70
MaxRequestSize connection parameter		ml_database	
enhancement in version 10.0.0	13	new feature in version 10.0.0	70
M-Business Anywhere		ml_delete_sync_state system procedure	
UltraLite enhancements in 9.0.2	138	MobiLink new feature in version 10.0.0	69
UltraLite new feature in 9.0.1	167	ml_delete_sync_state_before system procedure	
M-Business Anywhere Redirector		MobiLink new feature in version 10.0.0	69
new in version 9.0.2	134	ml_global script version	
MDSR encryption		MobiLink new feature in version 10.0.0	72
discontinued in version 9.0.1	172	ml_listening	
Mem Pages Carver statistic		new schema in version 10.0.0	70
new feature in version 10.0.0	12	ml_qa_delivery	
Mem Pages Pinned Cursor statistic		new QAnywhere MobiLink system table in 9.0.2	
new feature in version 10.0.0	12	143	
Mem Pages Query Processing statistic		new schema in version 10.0.0	70
new feature in version 10.0.0	12	ml_qa_delivery_client	
merge joins		new QAnywhere MobiLink system table in 9.0.2	
new feature in version 8.0.0	241	143	
Message Agent		ml_qa_global_props	
new feature in version 9.0.2	137	changed in version 10.0.0	70
message link options		ml_qa_repository	
new SQL Remote features in version 6.0.3	316	new schema in version 10.0.0	70
message links		ml_qa_repository_content	
SQL Remote behavior change in version 6.0.3320		QAnywhere MobiLink system table removed in	
message logging		9.0.2	144
new SQL Remote feature in version 6.0.2	324	ml_qa_repository_props	
MESSAGE statement		changed in version 10.0.0	70
enhancement in version 10.0.0	30	ml_qa_repository_staging_client	
enhancement in version 9.0.1	156	QAnywhere MobiLink system table removed in	
new features in version 6.0.1	330	9.0.2	144
messaging		ml_qa_status_staging_client	
MobiLink new feature in 9.0.1	164	QAnywhere MobiLink system table removed in	
migrate API wizard		9.0.2	144
UltraLite using	357	ml_remote_id option	
migrating databases		new in version 10.0.0	74

ml_reset_sync_state system procedure			
MobiLink new feature in version 10.0.0	69		
ml_script			
new schema in version 10.0.0	70		
ml_sis_sync_state			
new feature in version 10.0.0	70		
ml_subscription			
MobiLink system table changed in 9.0.0	195		
new schema in version 10.0.0	70		
ml_user			
MobiLink system table changed in 9.0.0	195		
new schema in version 10.0.0	70		
mlxtract			
removed in version 10.0.0	88		
mlxtract utility			
new feature in version 7.0.0	305		
mobile web services			
new feature in version 10.0.0	91		
MobiLink			
upgrading	342		
MobiLink behavior changes			
version 10.0.0	80		
version 9.0.2	148		
MobiLink file transfers			
MobiLink new feature in version 10.0.0	75		
MobiLink Monitor			
enhancements in version 10.0.0	73		
new feature in 8.0.2	219		
MobiLink new features			
version 10.0.0	68		
version 9.0.2	134		
MobiLink plug-in			
enhancement in version 10.0.0	114		
MobiLink synchronization server			
behavior changes in version 8.0.0	273		
upgrading	344		
MobiLink system tables			
behavior change in version 7.0.0	310		
new tables in 9.0.1	164		
upgrading	343		
MobiLink user names			
behavior change in version 10.0.0	87		
behavior change in version 8.0.0	273		
modify_last_download_timestamp			
new feature in 8.0.1	232		
modify_next_last_download_timestamp			
new feature in 8.0.1	232		
monitoring schema upgrades			
UltraLite enhancement in 9.0.1	169		
MONTHS function			
behavior change in version 6.0.3	319		
MultiByteCharSet property			
new feature in version 6.0.3	314		
multi-database support			
UltraLite new feature in 9.0.1	169		
MultiPageAllocs property			
new feature in version 10.0.0	10		
multiple result sets			
Interactive SQL behavior change in version 10.0.0	116		
multiple versions			
Adaptive Server Anywhere	341		
UltraLite	349		
multi-threaded applications			
new UltraLite feature in 8.0.1	233		
N			
named constraints			
new feature in 9.0.0	182		
Named Pipes			
unsupported in version 10.0.0	61		
named script parameters			
MobiLink new feature in version 10.0.0	71		
native Apache Redirector			
new in version 9.0.2	134		
Native UltraLite for Java			
UltraLite enhancements in 9.0.2	139		
Native UltraLite for Java API support			
UltraLite removed in version 10.0.0	110		
NCHAR data type			
new feature in version 10.0.0	5		
NcharCharSet property			
new feature in version 10.0.0	10		
nested loops anti-semijoin			
new feature in 8.0.2	213		
nested loops semijoin			
new feature in 8.0.2	213		
.NET support			
MobiLink new feature in 8.0.2	218		
NetBios protocol			
unsupported in version 8.0.0	271		
NetWare			
new feature in version 9.0.1	158		
server behavior change in version 10.0.0	62		

upgrading databases to version 10.0.0	336	Notifiers	
version 4.10 unsupported in version 8.0.0	271	MobiLink new feature in 9.0.0	194
network communication parameters		NSAPI Redirector	
new feature in 8.0.2	215	Solaris support added in version 9.0.2	134
network layer		NULL	
MobiLink improved feature in version 10.0.0	71, 104	behavior change in version 9.0.1	174
network protocols		empty string behavior change in version 8.0.0	269
MobiLink revised feature in version 10.0.0	78	NULLIF function	
UltraLite revised feature in version 10.0.0	99	new feature in version 6.0.1	330
network_connect_timeout option		NUMBER function	
MobiLink client protocol option replaced in		behavior change in version 8.0.0	268
version 10.0./0	75	NumLogicalProcessors property	11
network_leave_open		NumLogicalProcessorsUsed property	11
MobiLink [dbmlsrv8] new feature in 8.0.2	220	NumPhysicalProcessors property	11
network_name		NumPhysicalProcessorsUsed property	11
MobiLink [dbmlsrv8] new feature in 8.0.2	220	NumProcessorsAvail property	
network_open_timeout		unsupported in version 10.0.0	61
MobiLink [dbmlsrv8] new feature in 8.0.2	220	NumProcessorsMax property	
new features		unsupported in version 10.0.0	61
version 10.0.0	1	NVFS	
version 6.0.1	328	UltraLite enhancement in version 10.0.0	101
version 6.0.2	322, 324		
version 6.0.3	312, 316	O	
version 7.0.0	296, 305	object-based data flow	
version 7.0.1	288, 289, 291	MobiLink new feature in version 10.0.0	69
version 7.0.2	282	ODBC	
version 8.0.0	240	UltraLite new feature in 9.0.1	168
version 8.0.1	228	Version 5 compatibility	381
version 8.0.2	212	ODBC configuration dialog	
version 9.0.0	178	behavior change in version 9.0.1	173
version 9.0.1	154	ODBC data sources	
version 9.0.2	126	new feature in version 9.0.2	130
version 7.0.3	278	ODBC driver managers	
new_row_cursor		new Unix feature in version 10.0.0	33
deprecated in version 9.0.0	206	ODBC drivers	
removed in version 10.0.0	81	MobiLink behavior changes in version 10.0.0	89
newsgroups		Oracle driver new in version 10.0.0	70
technical support	xiv	-odbc option	
NextScheduleTime property		not supported for dbconsole in version 10.0.0	117
new feature in version 10.0.0	24	not supported for Interactive SQL in version	
non-synchronizing tables		10.0.0	117
new UltraLite feature in version 7.0.1	292	odbc_describe_binary_as_varbinary option	
Norwegian		new feature in version 9.0.2	128
new feature in version 10.0.0	15	odbc_distinguish_char_and_varchar option	
NoSyncOnStarup dbmlsync extended option		new feature in version 9.0.1	164
new feature in 10.0.0	76	oem_string option	
		new feature in version 10.0.0	9

OEM.ini file			
new feature in version 10.0.0	120	user-defined behavior change in version 8.0.0	266
old_row_cursor		output redirection	
deprecated in version 9.0.0	206	new features in version 7.0.0	303
removed in version 10.0.0	81	OUTPUT statement	
OLE DB		enhancement in version 9.0.0	183
behavior change in version 10.0.0	44	enhancement in version 9.0.1	160
OLE DB provider		new features in version 8.0.0	247
new feature in version 7.0.0	298	-oy option	
-on option		MobiLink [mlsrv10] removed in version 10.0.084	
MobiLink [dbmlsrv9] new feature in 9.0.0	195		
one-button sync		P	
new UltraLite for M-Business Anywhere feature		packet size	
in 9.0.2	138	behavior change in version 6.0.2	325
online books		behavior change in version 8.0.0	271
PDF	171	PacketSize property	
online help		new feature in version 8.0.0	250
new features in version 8.0.0	254	page sizes	
OPEN statement		behavior change in version 6.0.3	319
behavior change in version 8.0.0	267	default changed in version 10.0.0	43
enhancement in version 10.0.0	3	new features in version 7.0.0	300
openxml system procedure		PageSize property	
behavior change in version 10.0.0	44	new features in version 7.0.0	302
enhancement in version 9.0.1	163	Palm Computing Platform	
new feature in 9.0.0	178	new UltraLite features in version 8.0.0	262
optimization_goal option		required code change for version 8.0.0	274
default setting changed in 8.0.2	223	version 2 unsupported in version 8.0.0	274
new feature in version 7.0.0	300	Palm HotSync Conduit installer utility	
new feature in version 8.0.0	242	UltraLite enhancement in version 10.0.0	101
optimization_workload option		Palm OS	
new feature in version 9.0.1	157	earliest supported version in 9.0.1	175
Oracle		UltraLite enhancement in version 10.0.0	99
MobiLink ODBC driver new in version 10.0.0	89	parallel backups	
ORDER BY and UPDATE statements		new feature in version 10.0.0	14
behavior change in version 9.0.1	174	parallel index scans	
ORDER BY clause		new feature in version 9.0.1	157
enhancement in 9.0.0	181	password hashing	
UltraLite enhancement in version 10.0.0	102	behavior change in version 10.0.0	37
-os option		UltraLite behavior change in version 10.0.0	99
behavior change in version 10.0.0	47	-password option	
-ot option		QAnywhere option deprecated in 9.0.2	151
new feature in version 10.0.0	10	passwords	
outer joins		behavior change in version 10.0.0	37
behavior change in version 10.0.0	60	behavior change in version 9.0.1	173
new Transact-SQL feature in version 7.0.0	302	restrictions removed in version 10.0.0	39
Transact-SQL behavior changes	363	SQL Remote behavior change in version 6.0.3320	
outer reference aggregate functions		UltraLite behavior change in version 10.0.0	99
		paths	

UltraLite application upgrades	356	new features in version 8.0.0	242
UltraLite database upgrades	350	PLAN function	
PBUF connection parameter		new feature in 8.0.2	213
new feature in version 8.0.0	253	platform	
-pc option		UltraLite enhancement in version 10.0.0	98
QAnywhere new feature in version 10.0.0	93	platforms	
PCTFREE		UltraLite removed in version 10.0.0	110
new feature in 8.0.1	228	PlatformVer property	
PDF		new feature in version 8.0.0	251
enhancements in 9.0.1	171	plug-in modules	
online books	171	UltraLite new feature in version 10.0.0	100
PERCENT_RANK function		Pocket PC	
new feature in version 9.0.1	154	2002 supported in 8.0.1	233
performance		-port option	
enhancements in version 10.0.0	6	QAnywhere option removed in version 10.0.0	95
new feature in 8.0.2	216	PORT protocol option	
new features in version 8.0.0	240	behavior change in version 10.0.0	45
new MobiLink features in version 8.0.0	257	post	
UltraLite application upgrades	355	MobiLink [dblsn] enhancement in 9.0.2	136
UltraLite database upgrades	349	PowerBuilder	
UltraLite enhancement in version 10.0.0	97	using with SQL Anywhere	376
upgrading database files	336	PRC Tools	
Performance Monitor		new UltraLite feature in version 7.0.1	292
enhancement in version 10.0.0	12	precautions	
new feature in 8.0.2	216	upgrading	335
new features in version 6.0.3	314	predicates	
Perl		UltraLite new feature version 10.0.0	103
new DBD::ASAny driver in version 9.0.1	158	prefetch option	
persistent		enhancement in version 9.0.2	128
MobiLink [HTTP, HTTPS] new feature in 8.0.2	219	PrefetchBuffer connection parameter	
personal server		behavior change in version 10.0.0	42
behavior change in version 9.0.0	201	enhancement in version 10.0.0	13
default TCP/IP port changed in version 10.0.0	43	new feature in version 8.0.0	253
PHP module		PrefetchOnOpen connection parameter	
behavior change in version 10.0.0	42	new feature in 8.0.2	215
enhancements in version 10.0.0	32	PrefetchRows connection parameter	
new feature in version 9.0.2	128	new feature in version 8.0.0	253
-pi option		prefix files	
MobiLink [dbmlsync] new feature in 8.0.2	219	enhancements in 9.0.2	139
ping utility [dbping]		prepare_for_download	
behavior change in version 10.0.0	42	new feature in version 7.0.1	291
enhancement in version 10.0.0	20	PreparedStatement.addBatch method	
new feature in version 6.0.2	322	new feature in version 10.0.0	31
pinned_cursor_percent_of_cache option		PreserveSource property	
new feature in 8.0.1	231	database property deprecated in version 10.0.0	67
plan caching		new feature in version 8.0.0	251
		primary key constraints	

behavior change in version 10.0.0	38	new feature in version 10.0.0	114
procedure_profiling server option		QAnywhere web services	
behavior change in version 10.0.0	45	new feature in version 10.0.0	91
ProcedurePages property		gastop utility	
new feature in version 8.0.0	251	QAnywhere Agent	143
procedures		-qn option	
enhancement in version 10.0.0	22	new feature in version 10.0.0	16
new feature in 8.0.2	213	-qp option	
ProfileFilterConn property		new feature in version 8.0.0	250
new feature in version 10.0.0	25	qualify_table_owner parameter	
progress offsets		new feature in version 8.0.0	255
behavior change in version 10.0.0	87	Query Editor	
properties		enhancement in version 9.0.1	154
behavior change in version 10.0.0	37	new feature in version 8.0.0	245
PROPERTY function		query execution plan	
behavior change in version 9.0.2	148	Native UltraLite for Java feature in 9.0.2	139
enhancement in version 10.0.0	24	query optimization	
property functions		new features in version 8.0.0	240
enhancement in version 10.0.0	24	new UltraLite feature in 9.0.2	140
protocol options		query plans	
enhancement in version 10.0.0	13	new features in version 8.0.0	246
enhancements in version 10.0.0	13	new UltraLite feature in version 8.0.0	262
PROWS connection parameter		query processing	
new feature in version 8.0.0	253	new features in version 8.0.0	240
proxy ports		QueryCachePages property	
new feature in version 10.0.0	14	new feature in version 8.0.0	250, 251
proxy table creation wizard		QueryHeapPages property	
enhancement in version 9.0.1	162	new feature in version 10.0.0	10, 11
publications		QueryLowMemoryStrategy property	
new MobiLink feature in version 8.0.0	256	new feature in version 8.0.0	250, 251
new UltraLite feature in version 8.0.0	261	quoted_identifier option	
-push option		behavior change in version 10.0.0	116
QAnywhere new feature in version 10.0.0	93		
-push_notifications option	see -push option		
QAnywhere option renamed in version 10.0.0	95		
Q		R	
QAnywhere		RAISERROR statement	
MobiLink new feature in 9.0.1	164	new features in version 8.0.0	244
upgrading databases and applications	348	RAND function	
QAnywhere behavior changes		behavior change in version 10.0.0	38
version 10.0.0	95	RANK function	
version 9.0.2	151	new feature in version 9.0.1	154
QAnywhere new features		-rb option	
version 10.0.0	91	QAnywhere option deprecated in 9.0.2	151
version 9.0.2	142	READ statement	
QAnywhere plug-in		enhancement in version 9.0.1	160
		read_authdn parameter	
		new feature in version 10.0.0	16
		read_password parameter	

new feature in version 10.0.0	16	REGR_INTERCEPT function	
read-only		new feature in version 9.0.1	154
write files not supported in version 10.0.0	62	REGR_R2 function	
read-only databases		new feature in version 9.0.1	154
behavior change in version 10.0.0	40	REGR_SLOPE function	
new feature in version 6.0.3	312	new feature in version 9.0.1	154
write files not supported in version 10.0.0	62	REGR_SXX function	
REAL data type		new feature in version 9.0.1	154
new MobiLink feature in version 6.0.3	317	REGR_SXY function	
new UltraLite feature in version 6.0.3	317	new feature in version 9.0.1	154
rebuilding		REGR_SYY function	
troubleshooting	339	new feature in version 9.0.1	154
rebuilding databases		release_version	
behavior change in version 9.0.2	148	new feature in 9.0.2	134
failed rebuilds	339	remember_last_statement	
required for version 10.0.0	36	new feature in version 8.0.0	249
ReceiveBufferSize connection parameter		remote data access	
enhancement in version 10.0.0	13	new feature in version 6.0.1	328
ReceivingTracingFrom property		remote databases	
new feature in version 10.0.0	12	upgrading	345
recovery		remote DBA permissions	
enhancement in version 10.0.0	15	behavior change in version 10.0.0	60
recursive union		remote IDs	
new feature in 9.0.0	180	MobiLink new feature in version 10.0.0	74
Redirector		remote server creation wizard	
Apache web server support in 9.0.1	167	enhancement in version 9.0.1	161
enhancements in version 10.0.0	74	enhancement in version 9.0.2	126
Java Redirector new feature in 8.0.2	220	REMOTEPWD field	
reference databases		behavior change in version 7.0.0	308
UltraLite upgrades of	350	RemoteTrunc property	
referencing aliases		new feature in version 7.0.0	302
define before referencing	286	REORGANIZE TABLE statement	
referential integrity		new feature in version 8.0.0	248
UltraLite behavior change in version 9.0.2	150	REPLACE function	
UltraLite new feature in version 10.0.0	103	new features in version 7.0.0	303
referential integrity actions		Replication Agent	
behavior change in version 6.0.3	319	enhancement in version 10.0.0	19
REFRESH MATERIALIZED VIEW statement		new features in version 8.0.0	255
new feature in version 10.0.0	27	report_error event	
REFRESH TRACING statement		new feature in version 7.0.0	306
new feature in version 10.0.0	4, 27	report_odbc_error	
REGR_AVGX function		new feature in version 7.0.1	291
new feature in version 9.0.1	154	ReqCountActive property	
REGR_AVGY function		new feature in version 10.0.0	10
new feature in version 9.0.1	154	ReqCountBlockContention property	
REGR_COUNT function		new feature in version 10.0.0	10
new feature in version 9.0.1	154	ReqCountBlockIO property	

new feature in version 10.0.0	10	results	
ReqCountBlockLock property		Interactive SQL enhancement in version 9.0.1	161
new feature in version 10.0.0	10	Sybase Central enhancement in version 9.0.1	161
ReqCountUnscheduled property		resuming failed downloads	
new feature in version 10.0.0	10	dbmlsync new feature in 9.0.1	165
ReqTimeActive property		RetryConnectionTimeout connection parameter	
new feature in version 10.0.0	10	new feature in version 10.0.0	13
ReqTimeBlockContention property		RetryConnectionTimeout property	
new feature in version 10.0.0	10	new feature in version 10.0.0	10
ReqTimeBlockIO property		return codes	
new feature in version 10.0.0	10	Interactive SQL new feature in version 7.0.2	282
ReqTimeBlockLock property		MobiLink [dbmlsync] enhancement in version	
new feature in version 10.0.0	10	9.0.0	196
ReqTimeUnscheduled property		return_date_time_as_string option	
new feature in version 10.0.0	10	new feature in 8.0.2	216
request log		new features in version 8.0.0	248
new feature in version 9.0.2	131	return_java_as_string option	
request logging		unsupported in version 10.0.0	40
enhancement in version 10.0.0	17	REVERSE function	
enhancement in version 9.0.1	160	new feature in version 10.0.0	24
new feature in 8.0.2	216	REWRITE function	
new feature in version 9.0.2	131	new argument ANSI in 8.0.1	229
request_timeout option		ROLLBACK statement	
new feature in version 10.0.0	9	UltraLite dynamic SQL new feature in 9.0.1	167
RequestFilterDB property		rollback_on_deadlock option	
new feature in version 10.0.0	25	new feature in version 9.0.2	133
request-level logging <i>see</i> request logging		ROLLUP operation	
RequestLogging property		new feature in 9.0.0	181
enhancement in version 10.0.0	26	new feature in version 9.0.1	154
RequestLogNumFiles property		row ordering	
new feature in version 9.0.2	131	behavior change in version 8.0.0	267
RequestsReceived property		ROW_NUMBER function	
new feature in version 10.0.0	10	new feature in version 9.0.1	154
RequestStatus property		row-wise partitioning for MobiLink ASA clients	
new feature in version 10.0.0	10	MobiLink new behavior in 9.0.1	175
RequestTiming property		RSA	
new feature in version 10.0.0	26	Beta: RSA_FIPS strong encryption not supported	
restartable downloads		123	
UltraLite new feature in 9.0.1	169	included with MobiLink for version 10.0.0	77
RESTORE DATABASE statement		included with SQL Anywhere for version 10.0.0	
behavior change in version 8.0.0	269	17	
enhanced support for variables in version 9.0.1	156	included with UltraLite for version 10.0.0	99
new feature in version 6.0.2	322	MobiLink new feature in 8.0.2	220
new feature in version 9.0.2	131	RSA encryption	
restore database wizard		UltraLite new feature in version 10.0.0	104
enhancement in version 9.0.1	162	RSA encryption for synchronization	
		new UltraLite features in 8.0.2	222

S		
-s option		
new backup utility option in version 9.0.1	159	
sa_check_commit system procedure		
new feature in version 7.0.0	303	
sa_column_stats system procedure		
new feature in version 10.0.0	22	
sa_conn_activity system procedure		
new feature in version 8.0.0	249	
sa_conn_info system procedure		
enhancement in version 10.0.0	26	
sa_conn_list system procedure		
new feature in version 10.0.0	22	
sa_conn_options system procedure		
new feature in version 10.0.0	22	
sa_conn_properties_by_conn system procedure		
unsupported in version 10.0.0	67	
sa_conn_properties_by_name system procedure		
unsupported in version 10.0.0	67	
sa_convert_ml_progress_to_timestamp system procedure		
new feature in version 10.0.0	76	
sa_convert_timestamp_to_ml_progress system procedure		
new feature in version 10.0.0	76	
sa_db_list system procedure		
new feature in version 10.0.0	22	
SA_DEBUG group		
new feature in version 6.0.2	325	
sa_describe_query system procedure		
new feature in version 10.0.0	22	
sa_disk_free_space system procedure		
new feature in 8.0.1	228	
sa_flush_statistics system procedure		
new feature in 8.0.1	228	
sa_get_request_profile		
new features in version 8.0.0	250	
sa_get_request_profile system procedure		
new feature in version 9.0.2	131	
sa_get_request_times		
new features in version 8.0.0	250	
sa_get_request_times system procedure		
new feature in version 9.0.2	131	
sa_get_server_messages system procedure		
new feature in 8.0.1	228	
sa_index_density system procedure		
new feature in version 8.0.0	249	
new feature in version 9.0.2	131	
sa_index_levels system procedure		
new feature in version 9.0.2	131	
sa_java_loaded_classes system procedure		
new feature in version 8.0.0	253	
sa_load_cost_model system procedure		
new feature in version 10.0.0	23	
sa_locks system procedure		
behavior change in version 10.0.0	38	
new feature in version 7.0.0	297	
sa_make_object system procedure		
new feature in 8.0.2	217	
sa_materialized_view_info system procedure		
new feature in version 10.0.0	22	
sa_migrate system procedure		
new feature in 8.0.1	230	
new feature in version 7.0.2	284	
sa_performance_diagnostics system procedure		
enhancement in version 10.0.0	26	
sa_procedure_profile system procedure		
enhancement in version 10.0.0	25	
sa_procedure_profile_summary system procedure		
enhancement in version 10.0.0	25	
sa_refresh_materialized_views system procedure		
new feature in version 10.0.0	22	
sa_remove_tracing_data system procedure		
new feature in version 10.0.0	4, 23	
sa_report_deadlocks system procedure		
new feature in version 9.0.2	129	
sa_reset_identity system procedure		
behavior change in version 10.0.0	38	
sa_rowgenerator system procedure		
new feature in version 9.0.2	133	
sa_save_trace_data system procedure		
new feature in version 10.0.0	4, 23	
sa_send_udp system procedure		
new feature in version 9.0.2	133	
sa_server_option system procedure		
enhancement in version 10.0.0	25	
enhancement in version 9.0.1	160	
new feature in version 6.0.3	314	
new feature in version 7.0.0	304	
new feature in version 9.0.2	131	
sa_set_http_option system procedure		
enhancement in version 10.0.0	34	
sa_set_soap_header system procedure		
new feature in version 10.0.0	34	

sa_set_tracing_level system procedure		MobiLink [dbmlsync] changes in 9.0.0	196
new feature in version 10.0.0	4, 23	new feature in 9.0.0	184
sa_snapshots system procedure		new feature in version 7.0.0	296
new feature in version 10.0.0	3	new MobiLink features in version 7.0.1	290
sa_split_list system procedure		schema	
new feature in version 10.0.0	23	UltraLite new feature in version 10.0.0	97
sa_table_fragmentation system procedure		UltraLite upgrade impact of	349
new feature in version 8.0.0	249	Schema Painter	
sa_transactions system procedure		enhancements in 9.0.2	141
new feature in version 10.0.0	3	UltraLite removed in version 10.0.0	110
sa_unload_cost_model system procedure		schema upgrades	
new feature in version 10.0.0	23	UltraLite enhancement in 9.0.1	169
sa_validate system procedure		scjview.exe	
behavior change in version 10.0.0	38	new feature in 9.0.0	193
new feature in version 9.0.1	157	ScoutSync	
new features in version 8.0.0	248	deprecated feature	237
sa_verify_password system procedure		no longer supported	208
new feature in version 9.0.2	133	removed in version 9.0	207
SACHARSET environment variable		script versions	
new feature in version 10.0.0	42	configurable in version 9.0.1	165
SADIR environment variable		scripted upload	
new feature in version 10.0.0	42	new in version 10.0.0	76
SALANG environment variable		scripts	
new feature in version 10.0.0	42	upgrading	346
SALOGDIR environment variable		section 508	
new feature in version 10.0.0	42	compliance	245
sample certificates		secure_feature_key option	
behavior change in version 9.0.2	147	new feature in version 10.0.0	18
sample database		SecureFeatures property	
behavior change in version 8.0.0	268	new feature in version 10.0.0	18
renamed in version 10.0.0	121	security	
SATMP environment variable		MobiLink enhancements in version 10.0.0	77
deprecated in version 8.0.0	273	new feature in version 9.0.2	132
new feature in version 10.0.0	42	new UltraLite feature in version 8.0.0	261
-sc option		new UltraLite features in 8.0.2	222
database server option removed in version 10.0.0		UltraLite behavior change in version 9.0.2	151
62		UltraLite enhancements in version 10.0.0	99
scattered reads		security protocol option	
behavior change in version 10.0.0	38	deprecated for HTTP for 9.0.2	148
new feature in 8.0.2	215	segments	
schedule creation wizard		new UltraLite feature in version 8.0.0	263
new feature in version 10.0.0	114	SELECT statement	
scheduled events		behavior change in version 7.0.2	286
behavior change in 8.0.2	223	enhancements in version 10.0.0	27, 28
scheduling		selectivity estimates	
behavior change in 8.0.2	223	user-defined behavior change in version 8.0.0	266
MobiLink [dbmlsync] change in 8.0.2	225	SendBufferSize connection parameter	

enhancement in version 10.0.0	13	new feature in version 9.0.2	132
SendingTracingTo property		ServerPort protocol option	
new feature in version 10.0.0	12	behavior change in version 10.0.0	45
SeparateCheckpointLog property		servers	
database property deprecated in version 10.0.0	67	name restrictions removed in version 10.0.0	39
new feature in version 8.0.0	251	service creation wizard	
SeparateForeignKeys property		enhancement in version 9.0.1	162
database property deprecated in version 10.0.0	67	service property sheet	
new feature in version 8.0.0	251	enhancement in version 9.0.1	162
sequence number		service utility [dbsvc]	
MobiLink system table schema change in version 10.0.0	70	behavior change in version 10.0.0	43
sequential table scans		behavior change in version 9.0.2	147
new features in version 8.0.0	242	enhancement in version 10.0.0	20, 33
serial protocol		new feature in version 7.0.1	288
MobiLink does not support as of 8.0.2	225	new feature in version 9.0.2	129
UltraLite does not support as of 8.0.2	226	services	
server administration requests		enhancement in version 10.0.0	20
QAnywhere new feature in version 10.0.0	94	enhancement in version 9.0.1	162
server enumeration utility [dblocate]		set address and type as client options	
behavior change in version 10.0.0	43	new feature in 8.0.1	232
enhancement in version 9.0.1	159	SET OPTION statement	
new feature in version 10.0.0	20	behavior change in version 10.0.0	117
server error codes		SET statement	
MobiLink behavior change in version 9.0	206	enhancement in version 10.0.0	3
server logging		new feature in 8.0.2	214
new features in version 8.0.0	250	SET_BIT function	
Server Messages window		new feature in version 10.0.0	24
enhancement in version 10.0.0	16	SET_BITS function	
server messages window		new feature in version 10.0.0	24
enhancement in version 10.0.0	15	set_cookie protocol option	
new feature in version 9.0.2	133	new MobiLink feature in 9.0.2	136
server name		new UltraLite feature in 9.0.2	141
behavior change in version 10.0.0	39	-sf option	
maximum length behavior change in version 6.0.3 320		new feature in version 10.0.0	18
restrictions removed in version 10.0.0	39	SHA256 algorithm for hashing	
server properties		new feature in version 10.0.0	26
behavior change in version 10.0.0	37	SiteScriptName	
server startup options dialog		no longer supported	205
new feature in version 9.0.2	133	-sk option	
server-initiated synchronization		new feature in version 10.0.0	18
additions in version 10.0.0	78	Smartphone 2002	
MobiLink new feature in 9.0.0	194	UltraLite new feature in 9.0.1	169
ServerName property		SMTP message type	
new feature in version 10.0.0	11	new feature in 8.0.1	233
ServerPort property		SMTP system procedures	
		new features in version 8.0.0	255
		SMTP/POP message type	

new NetWare feature in version 7.0.1	291	new feature in version 8.0.0	255
new SQL Remote feature in version 6.0.3	316	sp_remote_tables system procedure	
snapshot isolation		new features in version 8.0.0	255
MobiLink support in version 10.0.0	71	Spanish	
new feature in version 10.0.0	3	new feature in version 9.0.2	129
SnapshotCount property		SPX	
new feature in version 10.0.0	10, 12	new feature in version 7.0.0	298
SNMP agent		SQL Anywhere	
new feature in 9.0.2	128	documentation	viii
SNMP extension agent		SQL Anywhere 10	
enhancement in version 10.0.0	17	upgrading to	333
SOAP clients		SQL Anywhere behavior changes	
new feature in version 9.0.2	128	version 10.0.0	36
SOAP servers		SQL Anywhere Broadcast Repeater utility [dbns10]	
new feature in version 9.0.2	128	new feature in version 10.0.0	20
SOAP_HEADER function		SQL Anywhere Console utility [dbconsole]	
new feature in version 10.0.0	35	enhancement in version 10.0.0	115
software compatibility		SQL Anywhere deprecated features	
UltraLite upgrades	349	version 10.0.0	60
software updates		SQL Anywhere discontinued features	
new feature in 9.0.0	192	version 10.0.0	60
sort_collation option		SQL Anywhere Explorer	
new feature in 8.0.1	230	new feature in version 10.0.0	31
SORTKEY function		SQL Anywhere new features	
new feature in version 8.0.0	244	version 10.0.0	3
sp_hook_dbmlsync_all_error		SQL Anywhere PHP module	
new in version 10.0.0	76	enhancements in version 10.0.0	32
sp_hook_dbmlsync_communication_error		new feature in version 9.0.2	128
new in version 10.0.0	76	SQL Anywhere plug-in	
sp_hook_dbmlsync_connect_failed		behavior change in version 10.0.0	118
new feature in 9.0.2	135	SQL Anywhere report submission utility (dbsupport)	
sp_hook_dbmlsync_download_com_error		new feature in version 10.0.0	21
deprecated in version 10.0.0	88	SQL Flagger	
sp_hook_dbmlsync_download_ri_violation		new feature in version 10.0.0	16
new feature in 8.0.2	218	SQL Remote	
sp_hook_dbmlsync_fatal_sql_error		ASE support deprecated	149
deprecated in version 10.0.0	88	new features in version 6.0.1	331
sp_hook_dbmlsync_misc_error		upgrading version 5 installations	359
new in version 10.0.0	76	SQL Remote behavior changes	
sp_hook_dbmlsync_schema_upgrade		version 10.0.0	96
new feature in version 9.0.0	196	version 9.0.2	149
sp_hook_dbmlsync_sql_error		SQL Remote for ASE	
deprecated in version 10.0.0	88	removed in version 10.0.0	96
new in version 10.0.0	76	SQL Remote new features	
sp_remote_exported_keys system procedure		version 10.0.0	96
new feature in version 8.0.0	255	version 9.0.2	137
sp_remote_imported_keys system procedure		SQL/XML	

new feature in 9.0.0	178	START AT clause	
SQLAnywhere PHP module		UltraLite enhancement in version 10.0.0	102
behavior change in version 10.0.0	42	start classes	
sqlanywhere_error function		MobiLink new feature in 8.0.2	218
new feature in version 10.0.0	32	START DATABASE statement	
sqlanywhere_errorcode function		enhancement in version 10.0.0	30
new feature in version 10.0.0	32	new features in version 7.0.0	301
sqlanywhere_execute function		start server in background utility [dbspawn]	
new feature in version 10.0.0	32	new features in version 6.0.3	316
sqlanywhere_identity function		new features in version 7.0.2	283
new feature in version 10.0.0	32	START SYNCHRONIZATION DELETE statement	
sqlanywhere_insert_id function		UltraLite enhancement in version 10.0.0	102
new feature in version 10.0.0	32	StartDBPermission property	
sqlanywhere_set_option function		new feature in version 10.0.0	11
enhancement in version 10.0.0	32	statement-based uploads	
SQL_DATABASE_NOT_FOUND error		new MobiLink feature in version 8.0.0	257
UltraLite behavior change in 9.0.0	209	static interfaces	
SQLLOCALE environment variable		defined	198
behavior change in version 8.0.0	270	Static Java API support	
unsupported in version 10.0.0	61	UltraLite removed in version 10.0.0	110
SQL Remote		statistical scripts	
upgrading	359	new MobiLink feature in version 8.0.0	259
sr_date_format		statistics	
new SQL Remote feature in version 6.0.3	317	behavior change in version 9.0.2	146
sr_time_format		STOP DATABASE statement	
new SQL Remote feature in version 6.0.3	317	new features in version 7.0.0	301
sr_timestamp_format		STOP ENGINE statement	
new SQL Remote feature in version 6.0.3	317	new features in version 7.0.0	301
ssqueue		STOP SYNCHRONIZATION DELETE statement	
deprecated in 9.0.2	149	UltraLite enhancement in version 10.0.0	102
removed in version 10.0.0	96	stored procedure result sets	
ssremote		new feature in 9.0.0	180
deprecated in 9.0.2	149	stored procedure source format	
removed in version 10.0.0	96	new feature in version 8.0.0	252
ssxtract		stored procedures	
removed in version 10.0.0	96	behavior change in version 8.0.0	267
stack size		enhancement in version 10.0.0	30
behavior change in version 6.0.1	332	new features in version 8.0.0	246
standard deviation		new Java features in version 6.0.3	315
new feature in 9.0.0	181	StreamCompression	
standard upgrade precautions	335	deprecated in version 8.0.0	274
standards		string_rtruncation option	
section 508 compliance	245	behavior change in version 10.0.0	39, 46
standards and compatibility		STRIP ON clause	
section 508 compliance	245	deprecated in version 10.0.0	61
START AT		strong encryption	
new feature in 9.0.0	181	new feature in version 8.0.0	243

new feature in version 9.0.2	132	synchronization parameters	
new UltraLite feature in version 8.0.0	260	UltraLite behavior change in version 9.0.2	151
STRTOUUID function		synchronization results	
behavior change in version 9.0.2	148	new UltraLite feature in 8.0.2	222
subqueries		synchronization scripts	
UltraLite new feature in 9.0.1	168	behavior change in version 7.0.0	310
subscriptions		synchronization sites	
new MobiLink feature in version 8.0.0	256	deprecated in version 8.0.0	273
support		synchronization stream	
newsgroups	xiv	UltraLite revised feature in version 10.0.0	99
support submission utility (dbsupport)		synchronization templates	
new feature in version 10.0.0	21	deprecated in version 8.0.0	273
supported platforms		synchronize_mirror_on_commit option	
UltraLite 9.0.0 changes	208	new feature in version 10.0.0	3, 9
suppress_tds_debugging option		syncsa.sql	
new feature in version 8.0.0	252	MobiLink new behavior in version 10.0.0	69
-sv option		SyncTrunc property	
QAnywhere option deprecated in 9.0.2	151	new feature in version 7.0.0	302
Swedish		SYSATTRIBUTE system table	
new feature in version 9.0.2	129	new feature in 8.0.1	228
Sybase Central		removed in version 10.0.0	58
enhancement in version 10.0.0	113	SYSATTRIBUTENAME system table	
managing older databases	334	new feature in 8.0.1	228
new features in version 8.0.0	245	removed in version 10.0.0	58
new features in version 9.0.1	162	SYSCOLLATION system view	
Sybase Central behavior changes		deprecated in version 10.0.0	57
version 10.0.0	116	SYSCOLLATIONMAPPINGS system view	
Sybase Central new features		deprecated in version 10.0.0	57
version 10.0.0	113	SYSCOLUMN system view	
SYNC gateway		deprecated in version 10.0.0	57
new server-initiated synchronization feature in		SYSCONSTRAINT system view	
version 10.0.0	78	behavior change in version 10.0.0	59
synchronization		SYSDEPENDENCY system view	
new MobiLink features in version 7.0.1	290	new in version 10.0.0	55
new UltraLite features in version 8.0.0	261	SYSEXTENT system table	
UltraLite behavior change in version 10.0.0	103	removed in version 10.0.0	58
synchronization callback functions		SYSEXTERNLOGINS system table	
new UltraLite feature in 9.0.2	141	renamed in version 10.0.0	58
synchronization definitions		SYSFKCOL system view	
deprecated in version 8.0.0	273	deprecated in version 10.0.0	57
synchronization error reporting		SYSFKEY system view	
new Native UltraLite for Java feature in 9.0.2	139	new in version 10.0.0	55
synchronization ID		SYSFOREIGNKEY system view	
MobiLink new feature in version 10.0.0	72	deprecated in version 10.0.0	57
synchronization observer		SYSHISTORY system table	
new UltraLite features in 8.0.2	222	new feature in version 9.0.1	160
UltraLite enhancement in 9.0.1	169	SYSHISTORY system view	

enhancement in version 10.0.0	14	new in version 10.0.0	55
SYSIDX system view		SYSPROXYTAB system view	
new in version 10.0.0	55	new in version 10.0.0	55
SYSIDXCOL system view		SYPUBLICATION system view	
new in version 10.0.0	55	sync_type column added in version 10.0.0	76
SYSINDEX system view		SYSSERVERS system table	
behavior change in 10.0.0	66	renamed in version 10.0.0	58
deprecated in version 10.0.0	57	SYSSOURCE system view	
SYSINFO compatibility view		new in version 10.0.0	55
behavior change in version 10.0.0	41	SYSSYNCSCRIPT system view	
SYSIXCOL system view		new in version 10.0.0	55
deprecated in version 10.0.0	57	SYSTAB system view	
SYSJAVACLASS system view		enhancement in version 10.0.0	16
behavior change in version 10.0.0	41	SYSTABCOL system view	
SYSLOGIN system table		new in version 10.0.0	55
removed in version 10.0.0	58	SYSTABLE system view	
SYSLOGINMAP system view		deprecated in version 10.0.0	57
new in version 10.0.0	55	system data sources	
SYSMVOPTION system view		new feature in version 7.0.2	284
new in version 10.0.0	55	system path	
SYSMVOPTIONNAME system view		UltraLite utilities	349
new in version 10.0.0	55	utilities	341
synccasa.sql		system tables	
name changed to syncsa.sql in 10.0.0	90	behavior change in version 10.0.0	47
SYSOBJECT system view		new feature in version 7.0.0	300
new in version 10.0.0	55	system triggers	
SYSOPTBLOCK system table		behavior change in version 6.0.3	319
removed in version 10.0.0	58	system views	
SYSOPTJOINSTRATEGIES system view		behavior change in version 10.0.0	47
removed in version 10.0.0	58	SYSUSER system view	
SYSOPTJOINSTRATEGY system table		new in version 10.0.0	55
removed in version 10.0.0	58	SYSUSERAUTHORITY system view	
SYSOPTORDER system table		new in version 10.0.0	55
removed in version 10.0.0	58	SYSUSERMESSAGES system table	
SYSOPTORDERS system view		renamed in version 10.0.0	58
removed in version 10.0.0	58	SYSUSERPERM system view	
SYSOPTQUANTIFIER system table		deprecated in version 10.0.0	57
removed in version 10.0.0	58	SYSUSERPERMS system view	
SYSOPTREQUEST system table		deprecated in version 10.0.0	57
removed in version 10.0.0	58		
SYSOPTREWRITE system table		T	
removed in version 10.0.0	58	table encryption	
SYSPHYSIDX system view		new feature in version 10.0.0	8
new in version 10.0.0	55	table hints	
SYSPROCEDURES view		new feature in 8.0.2	215
removed in version 10.0.0	58	table order	
SYSPROCS system view		UltraLite new feature in version 10.0.0	104

table owners			
new UltraLite feature in 9.0.2	140	new MobiLink client protocol option in 10.0.0	75
table scans		timestamp_format option	
new features in version 8.0.0	242	behavior change in version 10.0.0	47
TableBitMaps property		new behavior in version 8.0.0	270
database property deprecated in version 10.0.0	67	timezone	
new feature in version 8.0.0	251	new features in version 8.0.0	244
TableOrderChecking dbmlsync extended option		TimeZoneAdjustment property	
new feature in version 10.0.0	77	new feature in version 8.0.0	244
task list		TLS	
new feature in version 10.0.0	113	UltraLite new feature in version 10.0.0	104
task scheduling		TOP clause	
new feature in version 7.0.0	296	enhanced support for variables in version 9.0.1	
TCP/IP		156	
behavior change in version 10.0.0	43, 45	new features in version 7.0.0	301
encryption in version 8.0.0	243	TRACED_PLAN function	
new features in version 7.0.0	298	new feature in version 10.0.0	4, 24
technical support		transaction log	
newsgroups	xiv	behavior change in version 9.0.2	147
temp_space_limit_check option		new feature in version 9.0.2	131
behavior change in version 10.0.0	47	transaction log offsets	
new feature in version 9.0.2	133	upgrading database files	336
temporary files		transaction log utility [dblog]	
enhancement in version 10.0.0	10	behavior change in version 9.0.2	147
new features in version 6.0.3	313	new features in version 6.0.3	316
temporary stored procedures		new SQL Remote feature in version 6.0.3	317
new feature in version 10.0.0	30	transaction logs	
temporary tables		enhancement in version 10.0.0	15
enhancements in version 10.0.0	30	transactions	
TempTablePages property		UltraLite dynamic SQL new feature in 9.0.1	167
new feature in version 8.0.0	250, 251	TransactionsSpanLogs property	
TEXT data type		database property deprecated in version 10.0.0	67
new MobiLink feature in version 7.0.0	305	new feature in version 8.0.0	251
storage enhancement in version 7.0.0	300	TransactionStartTime property	
Thai		new feature in version 9.0.1	163
new feature in version 9.0.2	129	Transact-SQL	
threading		outer join behavior changes	363
new features in version 6.0.1	330	transport-layer security	
new UltraLite Java feature in 8.0.2	221	new feature in version 9.0.2	132
threads		new features in version 7.0.1	292
new Java features in version 6.0.3	315	new MobiLink feature in version 7.0.1	289
new MobiLink feature in version 7.0.2	285	where to obtain certificates	225
three-tiered computing		Treo 600	
new feature in version 7.0.0	297	support added in version 10.0.0	80
time_zone_adjustment option		Treo 650	
new feature in version 8.0.0	244	support added in version 10.0.0	80
timeout option		TRIGGER EVENT statement	
		new feature in version 7.0.0	296

triggers			
behavior change in version 8.0.0	268	ULEnableFileDB function	
new feature in 8.0.2	216	behavior change in version 8.0.0	274
troubleshooting		ULEnableGenericSchema function	
command line utilities	341	deprecated feature in 9.0.1	175
define alias before referencing it	286	new UltraLite feature in 8.0.2	221
new features in version 7.0.2	283	ULEnablePalmRecordDB function	
UltraLite command line utilities	349	behavior change in version 8.0.0	274
upgrading databases to version 10	339	ulgen utility	
TRUNCATE TABLE statement		behavior change in version 8.0.0	274
performance enhancement in version 7.0.0	304	removed in version 10.0.0	110
truncate_date_values option		UltraLite database upgrades for	356
deprecated in version 9.0.2	146	ULGetRowCount function	
truncate_timestamp_values option		new UltraLite feature in version 8.0.0	264
new feature in version 7.0.2	284	ulinfo utility	
-tu option		UltraLite new feature in version 10.0.0	101
dbmlsync new feature in 9.0.1	165	ulinit utility	
typing completion		UltraLite enhancement in version 10.0.0	102
new feature in version 10.0.0	114	upgrading databases with	354
		ulisql utility	
		removed in version 10.0.0	110
		ulload utility	
		new UltraLite feature in 9.0.2	141
		UltraLite enhancement in version 10.0.0	102
		upgrading databases with	352
		ULPalmExit function	
		deprecated feature in 9.0.1	175
		ULPalmLaunch function	
		deprecated feature in 9.0.1	175
		ulpalmos.h	
		prefix file enhancement in 9.0.2	139
		ULRegisterErrorCallback function	
		UltraLite C/C++ new feature in 9.0.1	168
		ULSecureCerticomTLSStream	
		UltraLite behavior change in version 9.0.2	151
		ULSQLCONNECT environment variable	
		UltraLite new feature in version 10.0.0	102
		ulsync utility	
		new UltraLite feature in 9.0.2	141
		UltraLite enhancement in version 10.0.0	102
		ULTable	
		ULGetRowCount function	264
		UltraLite	
		new feature in version 6.0.2	322
		platform support changes in 9.0.0	208
		upgrading databases and applications	349
		UltraLite applications	
		upgrade paths	356

UltraLite behavior changes		UltraLite Static Java API	
version 10.0.0	110	deprecated in 9.0.2	150
version 9.0.2	150	UltraLite synchronization utility	
UltraLite components		UltraLite enhancement in version 10.0.0	102
new feature in 8.0.2	221	UltraLite unload old database utility	
table API behavior change in 9.0.1	176	UltraLite new feature in version 10.0.0	101
UltraLite database converter		UltraLite unload XML to database utility	
UltraLite new feature in 9.0.1	169	UltraLite enhancement in version 10.0.0	102
UltraLite database creation utility		UltraLite.NET	
UltraLite enhancement in version 10.0.0	102	ADO.NET interface in version 9.0.2	138
UltraLite databases		ADO.NET interface recommended	150
connection approaches	349	enhancements in 9.0.2	139
UltraLite enhancement in version 10.0.0	97, 105	UltraLite.NET component interface deprecated in	
UltraLite new feature in version 10.0.0	98	9.0.2	150
upgrade file format	349	UltraLite.NET controls	
upgrade paths	350	new feature in 9.0.1	167
UltraLite embedded SQL		ulunload utility	
deprecated in 9.0.2	150	new UltraLite feature in 9.0.2	141
UltraLite engine		UltraLite enhancement in version 10.0.0	102
UltraLite for M-Business Anywhere support in		ulunloadold utility	
version 9.0.2	138	UltraLite new feature in version 10.0.0	101
UltraLite new feature in 9.0.1	168	upgrading databases with	352
UltraLite for M-Business Anywhere		ulview utility	
new feature in 9.0.1	167	removed in version 10.0.0	110
UltraLite generator		ulxml utility	
behavior change in version 8.0.0	274	removed in version 10.0.0	110
new features in version 8.0.0	261	uncompress database wizard	
new UltraLite features in 8.0.2	222	unsupported in version 10.0.0	63
UltraLite information utility		Unicode	
UltraLite new feature in version 10.0.0	101	UltraLite upgrade recommendation for	349
UltraLite initialize database utility		UNION operator	
UltraLite enhancement in version 10.0.0	102	UltraLite enhancement in version 10.0.0	102
UltraLite Interactive SQL		unique identifiers	
new feature in 9.0.2	141	enhancement in version 10.0.0	32
UltraLite load XML to database utility		new feature in 8.0.2	212
UltraLite enhancement in version 10.0.0	102	UNIQUEIDENTIFIER data type	
UltraLite new features		enhancement in 9.0.2	126
version 10.0.0	97	enhancement in version 10.0.0	32
version 9.0.2	138	Unix	
UltraLite Plan tab		behavior change in version 10.0.0	44
removed in version 10.0.0	117	new features in version 10.0.0	33
UltraLite plug-in		unload data dialog	
new feature in version 10.0.0	114	new feature in version 9.0.2	130
UltraLite Static C++ API		unload database wizard	
deprecated in 9.0.2	150	behavior change in version 10.0.0	45
UltraLite static interfaces		using	337
terminology	198	UNLOAD statement	

enhanced support for variables in version 9.0.1			
156			
new feature in version 7.0.0	297		
UNLOAD TABLE statement			
enhanced support for variables in version 9.0.1			
156			
enhancements in version 10.0.0	28		
unload utility [dbunload]			
behavior change in version 10.0.0	41, 43		
deprecated feature in version 9.0.1	172		
enhancement in version 10.0.0	21		
enhancement in version 9.0.1	159		
new features in version 6.0.3	313		
upgrading database files	339		
unloading			
enhancement in version 10.0.0	21		
unloading version 5.x databases from Sybase Central			
new feature in 8.0.1	230		
unloading version 6.x databases from Sybase Central			
new feature in 8.0.1	230		
uo dbmlsync extended option			
MobiLink new feature in 9.0.0	196		
-uo option			
MobiLink [dbmlsync] new feature in 9.0.0	196		
MobiLink new feature in 9.0.0	196		
UPDATE statement			
new Java feature in version 6.0.3	315		
updating			
new feature in 8.0.2	212		
upgrade database utility [dbupgrad]			
behavior change in version 10.0.0	36		
upgrade database utility [ulunloadold]			
upgrading databases with	352		
upgrade database wizard			
behavior change in version 10.0.0	36		
enhancement in version 9.0.1	162		
new feature in 8.0.1	230		
UltraLite using	351		
upgrade utility [dbupgrad]			
behavior change in version 10.0.0	41		
upgrades			
compatibility library	378		
databases	386		
dbclient	384		
standalone applications	368		
upgrading			
about upgrading to version 10	333		
behavior change in version 10.0.0	36		
database file format	336		
MobiLink	342		
MobiLink consolidated databases	343		
MobiLink synchronization server	344		
MobiLink system tables	343		
new UltraLite feature in 8.0.2	221		
precautions	335		
QAnywhere	348		
remote databases	345		
SQL Anywhere	334		
SQL Remote version 5	359		
SQL Remote version 5	387		
SQL Remote	359		
synchronization scripts	346		
UltraLite application code	355		
UltraLite application paths	356		
UltraLite database file format	349		
UltraLite database paths	350		
UltraLite overview for databases and applications	349		
version 5 components	362		
version 5 databases	363		
upgrading applications			
UltraLite process for	355		
UltraLite upgrade paths	356		
upgrading databases			
NetWare	336		
old file formats deprecated in version 9.0.2	146		
restrictions	336		
UltraLite considerations for	349		
UltraLite upgrade paths	350		
upgrading SQL Anywhere			
about	333		
upgrading to version 10.0.0			
about	333		
upload stream			
new MobiLink feature in version 7.0.2	285		
new MobiLink features in version 8.0.0	258		
upload_cursor			
deprecated in version 9.0.0	206		
removed in version 10.0.0	81		
upload_deleted_rows			
new behavior in version 10.0.0	84		
upload_fetch_column_conflict			
MobiLink new feature in version 10.0.0	71		
upload_inserted_rows			

new behavior in version 10.0.0	84	new feature in 8.0.2	217
upload_updated_rows		UltraLite generation [ulgen] utility	356
new behavior in version 10.0.0	84	utility database	
UploadOnly dbmlsync extended option		new feature in version 6.0.3	314
MobiLink new feature in 9.0.0	196	uuid_has_hyphens option	
upload-only synchronization		new feature in version 10.0.0	9
MobiLink new feature in 9.0.0	196	UUIDs	
new UltraLite feature in version 7.0.1	293	MobiLink new feature in 8.0.2	218
-urc option		new feature in 8.0.2	212
MobiLink [dbmlsync] new feature in 8.0.2	219	new UltraLite feature in 8.0.2	221
-us option		-ux option	
dbmlsync new feature in 9.0.1	165	MobiLink [dbmlsync] new feature in 10.0.0	74
MobiLink [mlsrv10] removed in version 10.0.081		MobiLink [mlsrv10] new feature in 10.0.0	74
use_cookies		new feature in version 9.0.2	133
MobiLink HTTP option removed in 8.0.2	226	SQL Remote [dbremote] new feature in 10.0.0	96
user authentication			
new MobiLink feature in version 7.0.1	289	V	
new MobiLink feature in version 8.0.0	258	-v option	
new UltraLite feature in version 8.0.0	260	database server behavior change in 6.0.1	332
new UltraLite Java feature in 8.0.1	233	MobiLink [dbmlsync] enhanced feature in 8.0.2	219
user data sources		V4T mode	
new feature in version 7.0.2	284	UltraLite new feature in 9.0.1	169
user ID mapping		VALIDATE authority	
new feature in 8.0.1	232	new feature in version 10.0.0	18
user IDs		VALIDATE CHECKSUM statement	
behavior change in version 10.0.0	39	new feature in version 9.0.1	157
name restrictions removed in version 10.0.0	39	VALIDATE DATABASE statement	
UserAppInfo property		new feature in version 10.0.0	27
new feature in version 9.0.1	163	validate database wizard	
user-defined functions		enhancement in version 9.0.1	162
behavior change in version 8.0.0	267	enhancement in version 9.0.2	130
enhancement in version 10.0.0	22	VALIDATE INDEX statement	
new Java features in version 6.0.3	315	enhancement in version 10.0.0	27
non-deterministic	231	enhancements in version 10.0.0	29
user-supplied selectivity estimates		new feature in version 7.0.0	297
behavior change in version 8.0.0	266	VALIDATE statement	
UTF-8		enhancement in version 10.0.0	27
UltraLite enhancement in version 10.0.0	99	VALIDATE statements	
UTF8 collation		behavior change in version 10.0.0	60
deprecated in version 10.0.0	61	VALIDATE TABLE statement	
UTF-8 encoding		new features in version 7.0.0	297
UltraLite upgrade recommendation for	349	validating	
UTF8BIN collation		behavior change in version 10.0.0	18
new feature in version 10.0.0	15	new feature in version 7.0.0	297
UtilCmdsPermitted property		new features in version 8.0.0	248
new feature in version 8.0.0	250	validation utility [dbvalid]	
utilities			

behavior change in version 9.0.1	173	new features	288, 289, 291
enhancement in version 9.0.1	157	version 7.0.2	
enhancements in version 10.0.0	21	behavior changes	286
ValuePtr parameter		new features	282
enhancement in version 10.0.0	3	version 7.0.3	
VARBIT data type		behavior changes	279
new feature in version 10.0.0	31	new features	278
VAREXISTS function		version 8.0.0	
new feature in 8.0.2	213	behavior changes	266, 273, 274
VariableHashSize property		new features	240
database property deprecated in version 10.0.0	67	new features in Adaptive Server Anywhere	240
new feature in version 8.0.0	251	new features in MobiLink	256
variables		new features in UltraLite	260
new feature in 8.0.2	217	version 8.0.1	
variance		behavior changes	235
new feature in 9.0.0	181	new features	228
-ve option		version 8.0.2	
MobiLink [dbmlsrv10] new feature in version		behavior changes	223
10.0.0	72	new features	212
-verbose option		version 9	
QAnywhere option deprecated in 9.0.2	151	behavior changes	201
VERIFY protocol option		new features	178
new feature in 8.0.1	230	version 9.0.1	
verify_password_function option		behavior changes	172
new feature in version 10.0.0	9	new features	154
VerifyServerName protocol option		version 9.0.2	
new feature in 8.0.1	230	behavior changes	146
Veritas Cluster Server agent		new features	126
new feature in version 10.0.0	16	Version Store Pages statistic	
version 10		new feature in version 10.0.0	12
upgrading to	333	versions	
version 10.0.0		upgrading synchronization scripts	346
behavior changes	1	VersionStorePages property	
new features	1	new feature in version 10.0.0	12
version 5		VFS	
upgrading SQL Remote installations	359	UltraLite enhancement in version 10.0.0	101
version 6.0.1		view dependencies	
behavior changes	332	enhancement in version 9.0.1	159
version 6.0.2		new feature in version 10.0.0	7
new features	322, 324	view matching	
version 6.0.3		new feature in version 10.0.0	5
behavior changes	320	Visual Basic .NET	
new features	316	UltraLite support for AppForge CrossFire	138
version 7.0.0		-vp option	
behavior changes	307, 309, 310	MobiLink [dbmlsrv9] new feature in 9.0.0	195
new features	305	-vw option	
version 7.0.1		MobiLink [dbmlsrv8] change in 8.0.2	225

VxWorks			
no longer supported	208	2.11 no longer supported	223, 226
W		dbmlsync exclusive lock by default	206
WAITFOR statement		dbmlsync new feature in 9.0.0 for loading dlls	196
new feature in version 9.0.2	127	new feature in 8.0.2	212
new features in version 8.0.0	244	new feature in version 10.0.0	14
warning messages		new feature in version 6.0.1	328
W prefix new feature in 9.0	198	new feature in version 9.0.2	133
web servers		new features in version 6.0.3	313
enhancement in version 10.0.0	33	new features in version 7.0.1	288
web service clients		SH3 chips no longer supported	223, 226
new feature in version 9.0.2	128	SH4 chips no longer supported	223, 226
web services		UltraLite upgrade consideration for	349
enhancement in version 10.0.0	35	Windows CE emulator	
new feature in version 9.0.2	128	new UltraLite feature in version 7.0.1	293
QAnywhere new feature in version 10.0.0	91	Windows groups	
webservice_namespace_host option		new feature in version 9.0.2	131
new feature in version 10.0.0	34	Windows Me	
what's new in version 10.0.0	1	unsupported in version 10	120
what's new in version 6.0.1	327	Windows NT	
what's new in version 6.0.2	321	unsupported in version 10	120
what's new in version 6.0.3	311	Windows Performance Monitor	
what's new in version 7.0.0	295	MobiLink support dropped in version 10.0.0	90
what's new in version 7.0.1	287	new feature in 8.0.2	216
what's new in version 7.0.2	281	Winsock	
what's new in version 7.0.3	277	enhancement in version 10.0.0	66
what's new in version 8.0.0	239	WITH (XLOCK) feature	
what's new in version 8.0.1	227	enhancement in version 9.0.1	158
what's new in version 8.0.2	211	WITH AUTO NAME	
what's new in version 9.0.0	177	new feature in 8.0.2	214
what's new in version 9.0.1	153	WITH HASH SIZE clause	
what's new in version 9.0.2	125	deprecated in version 8.0.0	272
wide characters		WITH table-hint	
UltraLite enhancement in version 10.0.0	106	new feature in 8.0.1	229
windowing functions		wizards	
new feature in version 9.0.1	154	UltraLite API migration	357
Windows		UltraLite new feature in version 10.0.0	100
behavior change in version 10.0.0	44	UltraLite upgrade database	351, 353
UltraLite upgrade consideration for	349	unload database	337
Windows 3.x		work tables	
unsupported in version 7.0.0	307	new behavior in version 8.0.0	241
Windows 95		new feature in 8.0.2	213
unsupported in version 10	120	write files	
Windows 98		unsupported in version 10.0.0	62
unsupported in version 10	120	.wrt	
Windows CE		file extension not supported in version 10.0.0	62

X

- x option
 - MobiLink [mlsrv10] new syntax in version 10.0.0 71
- XML
 - new feature in version 8.0.0 247
 - new features in 9.0.0 178
- XML data type
 - new feature in 9.0.0 178
- XMLAGG function
 - new feature in 9.0.0 178
- XMLCONCAT function
 - new feature in 9.0.0 178
- XMLELEMENT function
 - new feature in 9.0.0 178
- XMLFOREST function
 - new feature in 9.0.0 178
- XMLGEN function
 - new feature in 9.0.0 178
- xp_cmdshell system procedure
 - behavior change in 8.0.2 223
 - behavior change in version 10.0.0 37
- xp_read_file system procedure
 - behavior change in version 10.0.0 37
 - new feature in version 7.0.0 301
- xp_scanf system procedure
 - behavior change in version 10.0.0 37
- xp_sendmail system procedure
 - enhancement in version 10.0.0 26
 - new features in version 8.0.0 255
- xp_sprintf system procedure
 - behavior change in version 10.0.0 37
- xp_startsmtp system procedure
 - new features in version 8.0.0 255
- xp_stopsmtp system procedure
 - new features in version 8.0.0 255
- xp_write_file system procedure
 - behavior change in version 10.0.0 37
 - new feature in version 7.0.0 301
- xs option
 - server option behavior change in version 9.0.2147

Y

- y option
 - database server support removed in version 10.0.0 62

- YEARS function
 - behavior change in version 6.0.3 319

Z

- za option
 - MobiLink [mlsrv10] removed in version 10.0.083
- zac option
 - MobiLink [dbmlsrv9] deprecated feature 206
- zd option
 - MobiLink [dbmlsrv] removed in version 9 206
- ze option
 - MobiLink [mlsrv10] removed in version 10.0.083
- zec option
 - MobiLink [dbmlsrv9] deprecated feature 206
- zl option
 - new feature in version 8.0.0 249
- zn option
 - new feature in version 9.0.2 131
- zus option
 - MobiLink [mlsrv10] new feature in version 10.0.0 72
- zw option
 - MobiLink [dbmlsrv8] new feature in 8.0.2 219
- zwd option
 - MobiLink [dbmlsrv8] new feature in 8.0.2 219
- zwe option
 - MobiLink [dbmlsrv8] new feature in 8.0.2 219