



MobiLink Synchronization Reference

Part number: DC50018-01-0900-01

Last modified: June 2003

Copyright © 1989–2003 Sybase, Inc. Portions copyright © 2001–2003 iAnywhere Solutions, Inc. All rights reserved.

No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of iAnywhere Solutions, Inc. iAnywhere Solutions, Inc. is a subsidiary of Sybase, Inc.

Sybase, SYBASE (logo), AccelaTrade, ADA Workbench, Adaptable Windowing Environment, Adaptive Component Architecture, Adaptive Server, Adaptive Server Anywhere, Adaptive Server Enterprise, Adaptive Server Enterprise Monitor, Adaptive Server Enterprise Replication, Adaptive Server Everywhere, Adaptive Server IQ, Adaptive Warehouse, AnswerBase, Anywhere Studio, Application Manager, AppModeler, APT Workbench, APT-Build, APT-Edit, APT-Execute, APT-Library, APT-Translator, ASEP, AvantGo, AvantGo Application Alerts, AvantGo Mobile Delivery, AvantGo Mobile Document Viewer, AvantGo Mobile Inspection, AvantGo Mobile Marketing Channel, AvantGo Mobile Pharma, AvantGo Mobile Sales, AvantGo Pylon, AvantGo Pylon Application Server, AvantGo Pylon Conduit, AvantGo Pylon PIM Server, AvantGo Pylon Pro, Backup Server, BayCam, Bit-Wise, BizTracker, Certified PowerBuilder Developer, Certified SYBASE Professional, Certified SYBASE Professional (logo), ClearConnect, Client Services, Client-Library, CodeBank, Column Design, ComponentPack, Connection Manager, Convoy/DM, Copernicus, CSP, Data Pipeline, Data Workbench, DataArchitect, Database Analyzer, DataExpress, DataServer, DataWindow, DB-Library, dbQueue, Developers Workbench, Direct Connect Anywhere, DirectConnect, Distribution Director, Dynamic Mobility Model, Dynamo, e-ADK, E-Anywhere, e-Biz Integrator, E-Whatever, EC Gateway, ECMAP, ECRTP, eFulfillment Accelerator, Electronic Case Management, Embedded SQL, EMS, Enterprise Application Studio, Enterprise Client/Server, Enterprise Connect, Enterprise Data Studio, Enterprise Manager, Enterprise Portal (logo), Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, eProcurement Accelerator, eremote, Everything Works Better When Everything Works Together, EWA, Financial Fusion, Financial Fusion (and design), Financial Fusion Server, Formula One, Fusion Powered e-Finance, Fusion Powered Financial Destinations, Fusion Powered STP, Gateway Manager, GeoPoint, GlobalFIX, iAnywhere, iAnywhere Solutions, ImpactNow, Industry Warehouse Studio, InfoMaker, Information Anywhere, Information Everywhere, InformationConnect, InstaHelp, InternetBuilder, iremote, iScript, Jaguar CTS, jConnect for JDBC, KnowledgeBase, Logical Memory Manager, M-Business Channel, M-Business Network, M-Business Server, Mail Anywhere Studio, MainframeConnect, Maintenance Express, Manage Anywhere Studio, MAP, MDI Access Server, MDI Database Gateway, media.splash, Message Anywhere Server, MetaWorks, MethodSet, ML Query, MobicATS, My AvantGo, My AvantGo Media Channel, My AvantGo Mobile Marketing, MySupport, Net-Gateway, Net-Library, New Era of Networks, Next Generation Learning, Next Generation Learning Studio, O DEVICE, OASIS, OASIS (logo), ObjectConnect, ObjectCycle, OmniConnect, OmniSQL Access Module, OmniSQL Toolkit, Open Biz, Open Business Interchange, Open Client, Open Client/Server, Open Client/Server Interfaces, Open ClientConnect, Open Gateway, Open Server, Open ServerConnect, Open Solutions, Optima++, Partnerships that Work, PB-Gen, PC APT Execute, PC DB-Net, PC Net Library, PhysicalArchitect, Pocket PowerBuilder, PocketBuilder, Power Through Knowledge, Power++, power.stop, PowerAMC, PowerBuilder, PowerBuilder Foundation Class Library, PowerDesigner, PowerDimensions, PowerDynamo, Powering the New Economy, PowerJ, PowerScript, PowerSite, PowerSocket, Powersoft, Powersoft Portfolio, Powersoft Professional, PowerStage, PowerStudio, PowerTips, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, QAnywhere, Rapport, Relational Beans, RepConnector, Replication Agent, Replication Driver, Replication Server, Replication Server Manager, Replication Toolkit, Report Workbench, Report-Execute, Resource Manager, RW-DisplayLib, RW-Library, S.W.I.F.T. Message Format Libraries, SAFE, SAFE/PRO, SDF, Secure SQL Server, Secure SQL Toolset, Security Guardian, SKILS, smart.partners, smart.parts, smart.script, SQL Advantage, SQL Anywhere, SQL Anywhere Studio, SQL Code Checker, SQL Debug, SQL Edit, SQL Edit/TPU, SQL Everywhere, SQL Modeler, SQL Remote, SQL Server, SQL Server Manager, SQL Server SNMP SubAgent, SQL Server/CFT, SQL Server/DBM, SQL SMART, SQL Station, SQL Toolset, SQLJ, Stage III Engineering, Startup.Com, STEP, SupportNow, Sybase Central, Sybase Client/Server Interfaces, Sybase Development Framework, Sybase Financial Server, Sybase Gateways, Sybase Learning Connection, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase Synergy Program, Sybase User Workbench, Sybase Virtual Server Architecture, SybaseWare, Syber Financial, SyberAssist, SybMD, SyBooks, System 10, System 11, System XI (logo), SystemTools, Tabular Data Stream, The Enterprise Client/Server Company, The Extensible Software Platform, The Future Is Wide Open, The Learning Connection, The Model For Client/Server Solutions, The Online Information Center, The Power of One, TradeForce, Transact-SQL, Translation Toolkit, Turning Imagination Into Reality, UltraLite, UltraLite.NET, UNIBOM, Unilib, Uninull, Unisep, Unistring, URK Runtime Kit for UniCode, Versacore, Viewer, VisualWriter, VQL, Warehouse Control Center, Warehouse Studio, Warehouse WORKS, WarehouseArchitect, Watcom, Watcom SQL, Watcom SQL Server, Web Deployment Kit, Web.PB, Web.SQL, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, XA-Server, and XP Server are trademarks of Sybase, Inc. or its subsidiaries.

Certicom, MobileTrust, and SSL Plus are trademarks and Security Builder is a registered trademark of Certicom Corp. Copyright © 1997–2001 Certicom Corp. Portions are Copyright © 1997–1998, Consensus Development Corporation, a wholly owned subsidiary of Certicom Corp. All rights reserved. Contains an implementation of NR signatures, licensed under U.S. patent 5,600,725. Protected by U.S. patents 5,787,028; 4,745,568; 5,761,305. Patents pending.

All other trademarks are property of their respective owners.

Contents

About This Manual	vii
SQL Anywhere Studio documentation	viii
Documentation conventions	xi
The CustDB sample database	xiii
Finding out more and providing feedback	xiv
 I MobiLink Reference	 1
1 MobiLink Synchronization Server Options	3
MobiLink synchronization server	4
2 MobiLink Synchronization Client	35
MobiLink synchronization client	36
dbmlsync options	40
3 Synchronization Events	83
Overview of MobiLink events	86
authenticate_parameters connection event	98
authenticate_user connection event	100
authenticate_user_hashed connection event	104
begin_connection connection event	107
begin_connection_autocommit connection event	109
begin_download connection event	110
begin_download_table event	112
begin_download_deletes table event	114
begin_download_rows table event	116
begin_publication connection event	118
begin_synchronization connection event	121
begin_synchronization_table event	123
begin_upload connection event	125
begin_upload_table event	127
begin_upload_deletes table event	129
begin_upload_rows table event	131
download_cursor cursor event	133
download_delete_cursor cursor event	136
download_statistics connection event	139
download_statistics_table event	142

end_connection connection event	145
end_download connection event	147
end_download table event	149
end_download_deletes table event	151
end_download_rows table event	153
end_publication connection event	155
end_synchronization connection event	158
end_synchronization table event	160
end_upload connection event	162
end_upload table event	164
end_upload_deletes table event	166
end_upload_rows table event	168
example_upload_cursor table event	170
example_upload_delete table event	171
example_upload_insert table event	172
example_upload_update table event	173
handle_error connection event	174
handle_odbc_error connection event	177
modify_last_download_timestamp connection event	180
modify_next_last_download_timestamp connection event	182
modify_user connection event	184
new_row_cursor cursor event (deprecated)	186
old_row_cursor cursor event (deprecated)	189
prepare_for_download connection event	192
report_error connection event	194
report_odbc_error connection event	196
resolve_conflict table event	199
synchronization_statistics connection event	202
synchronization_statistics table event	205
time_statistics connection event	207
time_statistics table event	209
upload_cursor cursor event (deprecated)	212
upload_delete table event	214
upload_fetch table event	216
upload_insert table event	218
upload_new_row_insert table event	220
upload_old_row_insert table event	222
upload_statistics connection event	224
upload_statistics table event	227
upload_update table event	231

4 SQL Statements


233

ALTER PUBLICATION statement	234
---------------------------------------	-----

ALTER SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]	236
ALTER SYNCHRONIZATION USER statement [MobiLink]	238
CREATE PUBLICATION statement	240
CREATE SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]	243
CREATE SYNCHRONIZATION USER statement [MobiLink]	245
DROP PUBLICATION statement	255
DROP SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]	256
DROP SYNCHRONIZATION USER statement [MobiLink]	257
START SYNCHRONIZATION DELETE statement [MobiLink]	258
STOP SYNCHRONIZATION DELETE statement [MobiLink]	260
5 Stored Procedures	261
Stored procedures to add or delete scripts	262
Client event-hook procedures	269
6 Utilities	299
ActiveSync provider installation utility	300
MobiLink stop utility	303
MobiLink client database extraction utility (deprecated)	304
MobiLink user authentication utility	308
Certificate reader utility	310
Certificate generation utility	311
7 MobiLink System Tables	315
Introduction	316
8 Data Type Conversions	323
Sybase Adaptive Server Enterprise	324
IBM DB2	326
Oracle	328
Microsoft SQL Server	330
9 Character Set Considerations	331
Character set considerations	332
10 ODBC Drivers	335
ODBC drivers supported by MobiLink	336
11 Deploying MobiLink Applications	337
Deployment overview	338
Deploying the MobiLink server	339
Deploying Adaptive Server Anywhere MobiLink clients	342
Deploying UltraLite MobiLink clients	344

II	Error and Warning Messages	345
12	MobiLink Communication Error Messages	347
	Communication error messages sorted by code	348
	Communication error messages sorted by message	352
	Communication error messages sorted by constant	356
	Communication error descriptions	362
13	MobiLink Synchronization Server Error Messages	399
	MobiLink synchronization server error messages sorted by code . .	400
	MobiLink synchronization server error messages sorted message . .	406
	MobiLink synchronization server error descriptions	412
14	MobiLink Synchronization Server Warning Messages	437
	MobiLink synchronization server warning messages sorted by code .	438
	MobiLink synchronization server warning messages sorted by mes- sage	443
	MobiLink synchronization server warning descriptions	448
	Index	469

About This Manual

Subject	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.
Audience	This manual is for users of Adaptive Server Anywhere and other relational database systems who wish to add synchronization or replication to their information systems.
Before you begin	 For a comparison of MobiLink with other synchronization and replication technologies, see “Replication Technologies” [<i>Introducing SQL Anywhere Studio</i> , page 19].

SQL Anywhere Studio documentation

The SQL Anywhere Studio 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 SQL Anywhere Studio documentation is available in a variety of forms: in an online form that combines all books in one large help file; as separate PDF files for each book; and as printed books that you can purchase. The documentation consists of the following books:

- ◆ **Introducing SQL Anywhere Studio** This book provides an overview of the SQL Anywhere Studio database management and synchronization technologies. It includes tutorials to introduce you to each of the pieces that make up SQL Anywhere Studio.
- ◆ **What's New in SQL Anywhere Studio** This book is for users of previous versions of the software. It lists new features in this and previous releases of the product and describes upgrade procedures.
- ◆ **Adaptive Server Anywhere Getting Started** This book is for people new to relational databases or new to Adaptive Server Anywhere. It provides a quick start to using the Adaptive Server Anywhere database-management system and introductory material on designing, building, and working with databases.
- ◆ **Adaptive Server Anywhere Database Administration Guide** This book covers material related to running, managing, and configuring databases and database servers.
- ◆ **Adaptive Server 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.
- ◆ **Adaptive Server Anywhere SQL Reference Manual** This book provides a complete reference for the SQL language used by Adaptive Server Anywhere. It also describes the Adaptive Server Anywhere system tables and procedures.
- ◆ **Adaptive Server Anywhere Programming Guide** This book describes how to build and deploy database applications using the C, C++, and Java programming languages. Users of tools such as Visual Basic and PowerBuilder can use the programming interfaces provided by those tools. It also describes the Adaptive Server Anywhere ADO.NET data provider.

- ◆ **Adaptive Server Anywhere Error Messages** This book provides a complete listing of Adaptive Server Anywhere error messages together with diagnostic information.
- ◆ **SQL Anywhere Studio Security Guide** This book provides information about security features in Adaptive Server Anywhere databases. Adaptive Server Anywhere 7.0 was awarded a TCSEC (Trusted Computer System Evaluation Criteria) C2 security rating from the U.S. Government. This book may be of interest to those who wish to run the current version of Adaptive Server Anywhere in a manner equivalent to the C2-certified environment.
- ◆ **MobiLink Synchronization User's Guide** This book describes how to use the MobiLink data synchronization system for mobile computing, which enables sharing of data between a single Oracle, Sybase, Microsoft or IBM database and many Adaptive Server Anywhere or UltraLite databases.
- ◆ **MobiLink Synchronization Reference** This book is a reference guide to MobiLink command line options, synchronization scripts, SQL statements, stored procedures, utilities, system tables, and error messages.
- ◆ **iAnywhere Solutions ODBC Drivers** This book describes how to set up ODBC drivers to access consolidated databases other than Adaptive Server Anywhere from the MobiLink synchronization server and from Adaptive Server Anywhere remote data access.
- ◆ **SQL Remote User's Guide** This book describes all aspects of the SQL Remote data replication system for mobile computing, which enables sharing of data between a single Adaptive Server Anywhere or Adaptive Server Enterprise database and many Adaptive Server Anywhere databases using an indirect link such as e-mail or file transfer.
- ◆ **SQL Anywhere Studio Help** This book includes the context-sensitive help for Sybase Central, Interactive SQL, and other graphical tools. It is not included in the printed documentation set.
- ◆ **UltraLite Database User's Guide** This book is intended for all UltraLite developers. It introduces the UltraLite database system and provides information common to all UltraLite programming interfaces.
- ◆ **UltraLite Interface Guides** A separate book is provided for each UltraLite programming interface. Some of these interfaces are provided as UltraLite components for rapid application development, and others are provided as static interfaces for C, C++, and Java development.

In addition to this documentation set, PowerDesigner and InfoMaker include their own online documentation.

Documentation formats SQL Anywhere Studio provides documentation in the following formats:

- ◆ **Online documentation** The online documentation contains the complete SQL Anywhere Studio documentation, including both 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 9 ► 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.

- ◆ **Printable books** The SQL Anywhere books are provided as a set of PDF files, viewable with Adobe Acrobat Reader.

The PDF files are available on the CD ROM in the *pdf_docs* directory. You can choose to install them when running the setup program.

- ◆ **Printed books** The complete set of books is available from Sybase sales or from eShop, the Sybase online store. You can access eShop by clicking How to Buy ► eShop at <http://www.ianywhere.com>.

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 upper case, 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.

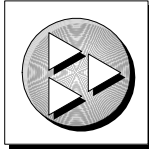
[**QUOTES** { **ON** | **OFF** }]

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

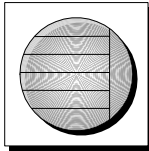
Graphic icons

The following icons are used in this documentation.

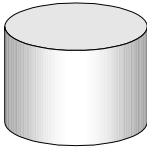
- ◆ A client application.



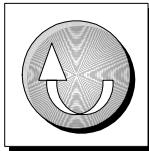
- ◆ A database server, such as Sybase Adaptive Server Anywhere.



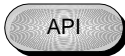
- ◆ A database. In some high-level diagrams, the icon may be used to represent both the database and the database server that manages it.



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



- ◆ A programming interface.



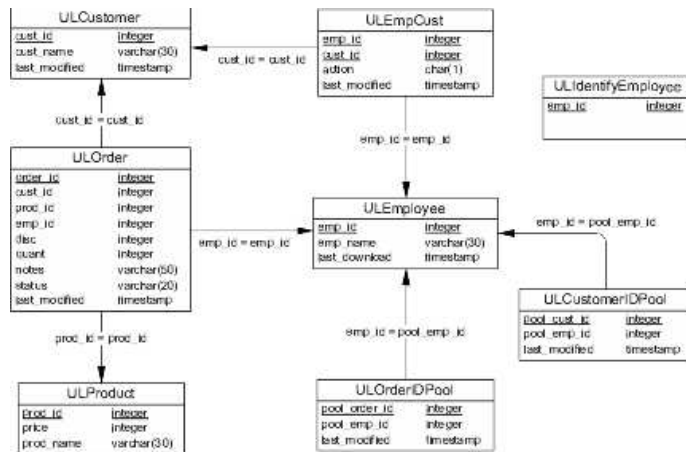
The CustDB sample database

Many of the examples in the MobiLink and UltraLite documentation use the UltraLite sample database.

The reference database for the UltraLite sample database is held in a file named *custdb.db*, and is located in the *Samples\UltraLite\CustDB* subdirectory of your SQL Anywhere directory. A complete application built on this database is also supplied.

The sample database is a sales-status database for a hardware supplier. It holds customer, product, and sales force information for the supplier.

The following figure shows the tables in the CustDB database and how they are related to each other.



Finding out more and providing feedback

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

You can provide feedback on this documentation and on the software through newsgroups set up to discuss SQL Anywhere technologies. These newsgroups can be found 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.

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 insure 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.

PART I

MOBILINK REFERENCE

This part contains reference material that describes in detail the various commands and components specific to MobiLink synchronization technology.

CHAPTER 1

MobiLink Synchronization Server Options

About this chapter

This chapter describes the options that can be set when starting the MobiLink synchronization server, dbmlsrv9.

Contents

Topic:	page
MobiLink synchronization server	4

MobiLink synchronization server

The MobiLink synchronization server lets you synchronize remote databases or applications with an ODBC-compliant consolidated database.

Function Start a MobiLink synchronization server.

Syntax **dbmlsrv9 -c "connection-string" [options]**

Option	Description
-a	Disable automatic reconnection upon synchronization error. See “-a option” on page 8 .
-b	Trim blank padding of strings. See “-b option” on page 8 .
-bc size	Specify the amount of memory to reserve for blob caching. See “-bc option” on page 9 .
-bn size	Specify the maximum number of bytes to consider when comparing blobs for conflict detection. See “-bn option” on page 10 .
-c "keyword=value; ..."	Supply ODBC database connection parameters for your consolidated database. See “-c option” on page 10 .
-cn connections	Set the maximum number of simultaneous connections with the consolidated database server. See “-cn option” on page 11 .
-cr count	Set the maximum number of database connection retries. See “-cr option” on page 11 .
-ct connection-timeout	Set the length of time a connection may be unused before it is timed out. See “-ct option” on page 11 .
-d number	Specify the size of the download cache. See “-d option” on page 11 .
-dl	Display all log messages on the console. See “-dl option” on page 12 .

Option	Description
-e filename	Store remote error logs sent into the named file. See “ -e option ” on page 12.
-et filename	Truncate the file and append remote synchronization logs to the new file. See “ -et option ” on page 12.
-f	Assume synchronization scripts do not change. See “ -f option ” on page 13.
-fr	If table data scripts are missing, synchronization will not abort, but just issue an error. See “ -fr option ” on page 13.
-o logfile	Log messages to a file. See “ -o option ” on page 13.
-on size	Set maximum size for log file. See “ -on option ” on page 14.
-oq	Prevent the popup dialog on startup error. See “ -oq option ” on page 15.
-os size	Maximum size of output file. See “ -os option ” on page 15.
-ot logfile	Log messages to a file, but truncate it first. See “ -ot option ” on page 16.
-ps num	Set maximum number of prepared statements to cache per connection. See “ -ps option ” on page 16.
-q	Minimize the synchronization server window. See “ -q option ” on page 16.
-r retries	Retry deadlocked uploads at most this many times. See “ -r option ” on page 16.
-rd delay	Set maximum delay, in seconds, before retrying a deadlocked transaction. See “ -rd option ” on page 17.
-s count	Specify the maximum number of rows to be fetched or sent at once. See “ -s option ” on page 17.

Option	Description
-sl dnet script-options	Set the .NET CLR options and force loading of the virtual machine on startup. See “ -sl dnet option ” on page 17.
-sl java script-options	Set the Java virtual machine options and force loading of the virtual machine on startup. See “ -sl java option ” on page 19.
-t ODBC-output-file	Log ODBC calls issued by MobiLink to this file. See “ -t option ” on page 19.
-tt ODBC-output-file	Log ODBC calls issued by MobiLink to this file, but first delete the file if it exists. See “ -tt option ” on page 20.
-u size	Specify the amount of memory to reserve for caching upload streams. See “ -u option ” on page 20.
-ud	On UNIX platforms, run as a daemon. See “ -ud option ” on page 20.
-v [levels]	Controls the type of messages written to the log file. See “ -v option ” on page 21.
-w count	Set the number of worker threads. See “ -w option ” on page 22.
-wu count	Set the maximum number of worker threads permitted to process uploads concurrently. See “ -wu option ” on page 23.
-x protocol[(network-parameters)]	Specify the communications protocol. Optionally, specify network parameters in form <i>parameter=value</i> , with multiple parameters separated by semicolons. See “ -x option ” on page 24.
-za	Allow generation of active scripts. See “ -za option ” on page 28.
-ze	Allow generation of sample scripts. See “ -ze option ” on page 29.

Option	Description
-zp	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. See “-zp option” on page 30.
-zs name	Specify a name for dbmlstop purposes. See “-zs option” on page 30.
-zt number	Specify the maximum number of processors used to run the MobiLink synchronization server. See “-zt option” on page 31.
-zu { + - }	Controls the automatic addition of users when the <code>authenticate_user</code> script is undefined. See “-zu option” on page 31.
-zw 1,...5	Controls which levels of warning message to display. See “-zw option” on page 31.
-zwd code	Disables specific warning codes. See “-zwd option” on page 32.
-zwe code	Enables specific warning codes. See “-zwe option” on page 33.

Description

The MobiLink synchronization server opens connections, via ODBC, with your consolidated database server. It then accepts connections from client applications and controls the synchronization process.

The MobiLink synchronization server is compatible with a variety of database-management systems, including Adaptive Server Anywhere, Adaptive Server Enterprise, Oracle, Microsoft SQL Server, and IBM DB2.

You must supply connection parameters for the consolidated database using the `-c` option. The command line options may be presented in any order. The `-c` option is shown here as the first item in a command string as a convention only. It can be anywhere in a list of options, but must precede a connection string.

Unless your ODBC data source is configured to automatically start the consolidated database, the database must be running before you start the MobiLink server.

You can put `dbmlsrv9` command line options in a configuration file and

optionally use the File Hiding utility, dbfhide, to add simple encryption to the configuration file.

☞ For more information, see “Using configuration files” [ASA *Database Administration Guide*, page 10] and “The File Hiding utility” [ASA *Database Administration Guide*, page 466].

dbmlsrv9 options

This section lists all MobiLink synchronization server command line options.

-a option

Function	Instructs the MobiLink synchronization server not to reconnect on synchronization error.
Syntax	dbmlsrv9 -c "connection-string" -a ...
Description	Should an error occur during synchronization, the MobiLink synchronization server automatically disconnects from the consolidated database, and then re-establishes the connection. Reconnecting ensures that the following synchronization starts from a known state. When this behavior is not required, you can use this option to disable it. The maintenance of state information depends on programmer requirements and may vary depending on the ways in which the programmer configures MobiLink scripting to work with the DBMS. This applies even if that database is an Oracle, Adaptive Server Anywhere database, or other supported product. Some status information may need to be re-initialized depending on the client application.

-b option

Function	For columns of type VARCHAR, CHAR, LONG VARCHAR, or LONG CHAR, removes trailing blanks from strings during synchronization.
Syntax	dbmlsrv9 -c "connection-string" -b ...
Description	<p>This option is intended to help resolve differences between the Adaptive Server Anywhere CHAR data type and the CHAR or VARCHAR data type used by the consolidated database. The Adaptive Server Anywhere CHAR data type is equivalent to VARCHAR. However, in most consolidated databases that are not Adaptive Server Anywhere, the CHAR(n) data type is blank-padded to n characters.</p> <p>When -b is specified, the MobiLink synchronization server removes trailing blanks from strings for columns of type CHAR, VARCHAR, LONG CHAR,</p>

or LONG VARCHAR if the column on the remote is a string. It does this before filtering rows that were uploaded in the current synchronization. The trimmed data is then downloaded to the remote databases.

This option can also be used to detect conflict updates. For each upload update row, the MobiLink synchronization server fetches the row from the consolidated database for the given primary key, compares the row with the pre-image of the update, and then determines whether the update is a conflict update. When -b is used, MobiLink trims trailing blanks from columns of type CHAR, VARCHAR, LONG CHAR, or LONG VARCHAR before doing the comparison.

Example

If the -b option is not used, a primary-key value of 'abc' uploaded from an Adaptive Server Anywhere or UltraLite remote to a CHAR(10) column in the consolidated database will become 'abc' followed by seven blank spaces. If the same row is downloaded, then it will appear on the remote as 'abc' followed by seven spaces. If the remote database is not blank-padded, then the remote will now have two rows: both 'abc' and 'abc' followed by seven spaces. There is now a duplicate row on the remote.

If the -b option is used, a primary-key value of 'abc' uploaded from an Adaptive Server Anywhere or UltraLite remote to a CHAR(10) column in the consolidated database will become 'abc' followed by seven spaces. Seven spaces still pad the value to ten characters, but if the same row is downloaded, then MobiLink server will strip the trailing spaces, and the value will appear on the remote as 'abc'. The -b option thus fixes the duplicate row problem.

-bc option

Function

Sets the blob cache size.

Syntax

dbmlsrv9 -c "connection-string" -bc size [k | m | g] . . .

Description

The amount of memory to use for caching blobs. If more memory is required, the MobiLink synchronization server uses disk space, instead. For this reason, too small a value can degrade performance. To calculate the minimum recommended size, multiply the maximum size of all blob data in any one row by the number of worker threads, then multiply the result by 4, which provides for a large reserve of memory.

The *size* is the amount of memory to reserve in bytes. Use the suffix k, m, or g to specify units of kilobytes, megabytes, or gigabytes, respectively. The default is 524 288 bytes.

-bn option

Function	Sets the maximum number of blob bytes to compare during conflict detection.
Syntax	dbmlsrv9 -c "connection-string" -bn size...
Description	<p>When two blobs contain similar or identical values, the operation of comparing them for filtering or conflict detection can be expensive due to the amount of data involved. This option tells the MobiLink synchronization server to consider only the first <i>size</i> bytes of two blobs when making the comparison. The default is to consider the entire blobs, no matter how big they are.</p> <p>Under some situations, limiting the maximum amount of data compared can speed synchronization substantially; however, it can also cause errors. For example, if two large blobs differ only in the last few bytes, the MobiLink synchronization server may consider them identical when in fact they are not.</p>

-c option

Function	Specifies connection parameters for the consolidated database.
Syntax	dbmlsrv9 -c "connection-string"...
Description	<p>The connection string must give the MobiLink synchronization server information sufficient to connect to the consolidated database. The connection string is required.</p> <p>The connection string must specify connection parameters in the form <i>keyword=value</i>, separated by semicolons, with no spaces between parameters.</p> <p>Connection parameters must be included in the ODBC data source specification if not given in the command line. Check your RDBMS and ODBC data source to determine required connection data.</p> <p>☞ For a complete list of SQL Anywhere connection parameters, see “Connection parameters” [ASA Database Administration Guide, page 174].</p> <p>☞ For information about how to hide the password, see “The File Hiding utility” [ASA Database Administration Guide, page 466].</p>

Example	<code>dbmlsrv9 -c "dsn=odbcname;uid=DBA;pwd=sql"</code>
---------	---

-cn option

Function	Sets the maximum number of simultaneous consolidated database connections.
Syntax	dbmlsrv9 -c "connection-string" -cn value...
Description	<p>Specifies the maximum number of simultaneous connections that the MobiLink synchronization server should make to the consolidated database. The minimum and the default value are one greater than the number of worker threads. A warning is issued if the supplied value is too small.</p> <p>A value larger than the number of worker threads may speed performance, particularly if connecting to the consolidated database is slow or if multiple script versions are in use. The optimum maximum number of database connections is the number of script versions times the number of worker threads, plus one. Connections above this optimum value will not necessarily speed synchronization, and will needlessly consume resources in both the MobiLink synchronization server and the consolidated database server.</p>

-cr option

Function	Sets the maximum number of database connection retries.
Syntax	dbmlsrv9 -c "connection-string" -cr value...
Description	Set the maximum number of times that the MobiLink synchronization server will attempt to connect to the database, before quitting, when a connection goes bad. The default value is three connection retries.

-ct option

Function	Sets the length of time, in minutes, that a connection may be unused before it is timed out and disconnected by the MobiLink synchronization server.
Syntax	dbmlsrv9 -c "connection-string" -ct connection-timeout...
Description	MobiLink database connections that go unused for a specified amount of time are freed by the server. The timeout can be set using the -ct option. A default timeout period of 60 minutes is used.

-d option

Function	Sets the size of the download cache.
Syntax	dbmlsrv9 -c "connection-string" -d number [k m g]...

Description	When no download acknowledgement is required, MobiLink buffers the download stream in a download cache. Since no acknowledgement is required from the client to commit the download transaction, the buffered download stream is sent to the client after the commit.
-------------	---

Use the suffix k, m, or g to specify units of kilobytes, megabytes, or gigabytes, respectively. The default size for the download cache is 0.5 megabytes.

-dl option

Function	Displays all log messages on screen.
----------	--------------------------------------

Syntax	dbmlsrv9 -c "<i>connection-string</i>" -v -dl ...
--------	--

Description	Display all log messages in the MobiLink synchronization server window. By default, only a subset of all messages is shown in the window when a log file is being output (using -o). In circumstances with many messages, this option can degrade performance.
-------------	--

-e option

Function	Stores error logs sent from Adaptive Server Anywhere MobiLink clients.
----------	--

Syntax	dbmlsrv9 -c "<i>connection-string</i>" -e <i>filename</i>...
--------	---

Description	With no -e option, error logs from Adaptive Server Anywhere MobiLink clients are stored in a file named <i>dbmlsrv.mle</i> . The -e option instructs the MobiLink synchronization server to store the error logs in the named file. By default, dbmlsync sends, on the occurrence of an error on the remote site, up to 32 kilobytes of remote log messages to a MobiLink synchronization server.
-------------	---

This option provides centralized access to remote error logs to help diagnose synchronization issues.

The amount of information delivered from a remote site can be controlled by the dbmlsync extended option `ErrorLogSendLimit`.

See also	“-et option” on page 12
----------	---

[“ErrorLogSendLimit \(el\) extended option” on page 50](#)

-et option

Function	Stores error logs sent from Adaptive Server Anywhere MobiLink clients in the named file after truncating the existing file.
----------	---

Syntax	dbmlsrv9 -c "<i>connection-string</i>" -et <i>filename</i>...
--------	--

Description The `-et` option is the same as the `-e` option, except that the error log file is truncated before any new errors are added to it.

The amount of information delivered from a remote site can be controlled by the `dbmlsync` extended option `ErrorLogSendLimit`.

See also [“ErrorLogSendLimit \(el\) extended option” on page 50](#)
[“-e option” on page 12](#)

-f option

Function Loads synchronization scripts only once, for better performance.

Syntax **dbmlsrv9 -c "connection-string" -f...**

Description Without the `-f` option, the MobiLink synchronization server checks to see if synchronization scripts have changed during regular operation. This checking is helpful during development, but can have an unnecessary performance impact in a production environment. With the `-f` option, the MobiLink synchronization server loads the synchronization scripts.

-fr option

Function If table data scripts are missing, synchronization will not abort, but just issue a warning.

Syntax **dbmlsrv9 -c "connection-string" -fr...**

Description Without the `-fr` option, the MobiLink synchronization server aborts if a synchronization does not include at least one script that uploads or downloads data. This option causes MobiLink to issue a warning instead of aborting.

-o option

Function Logs output messages to a message log file.

Syntax **dbmlsrv9 -c "connection-string" -o logfile...**

Description Write all log messages to the specified file. Note that the MobiLink synchronization server window, if present, usually shows a subset of all messages logged.

The MobiLink synchronization server gives the full error context in its output file if errors occur during synchronization. The error context may include the following information:

- ◆ **User Name** This is the actual user name that is provided by MobiLink

Adaptive Server Anywhere applications during synchronization.

- ◆ **Modified User Name** This is the user name as modified by the `modify_user` script.
- ◆ **Transaction** This lists the transaction the error occurs in. The transaction could be `authenticate_user`, `begin_synchronization`, `upload`, `prepare_for_download`, `download`, or `end_synchronization`.
- ◆ **Table Name** This shows the table name if it is available or NULL.
- ◆ **Row Operation** The operation could be INSERT, UPDATE, DELETE or FETCH.
- ◆ **Row Data** This shows all the column values of the row that caused the error.
- ◆ **Script Version** This is the script version currently used for synchronization.
- ◆ **Script** This is the script that caused the error.

Contextual information appears in the log regardless of your chosen level of verbosity.

See also

[“-os option” on page 15](#)

[“-ot option” on page 16](#)

[“-on option” on page 14](#)

[“-v option” on page 21](#)

-on option

Function	Specifies a maximum size for the MobiLink synchronization server message log file, after which the file is renamed with the extension <code>.old</code> and a new file is started.
Syntax	dbmlsrv9 -c "connection-string" -on size [k m] . . .
Description	<p>The <i>size</i> is the maximum file size for the output log, in bytes. Use the suffix <code>k</code> or <code>m</code> to specify units of kilobytes or megabytes, respectively. The minimum size limit is 10 Kb.</p> <p>When the log file reaches the specified size, the MobiLink synchronization server renames the output file with the extension <code>.old</code>, and starts a new one with the original name.</p>

Note

If the .old file already exists, it is overwritten. To avoid losing old log files, use the -os option instead.

This option cannot be used with the -os option.

See also

[“-o option” on page 13](#)

[“-os option” on page 15](#)

[“-ot option” on page 16](#)

[“-v option” on page 21](#)

-oq option

Function

On Windows, prevents the appearance of the error dialog when a startup error occurs.

Syntax

dbmlsrv9 -c "connection-string" -oq . . .

Description

By default, the MobiLink synchronization server displays a message box dialog if a startup error occurs. The -oq option prevents this dialog from being displayed.

-os option

Function

Sets the maximum size of the message log file, after which a new log file with a new name is created and used.

Syntax

dbmlsrv9 -c "connection-string" -os size [k | m] . . .

Description

The *size* is the maximum file size for logging output messages. The default units is bytes. Use the suffix k or m to specify units of kilobytes or megabytes, respectively. The minimum size limit is 10 kb.

Before the MobiLink synchronization server logs output messages to a file, it checks the current file size. If the log message will make the file size exceed the specified size, the MobiLink synchronization server renames the message log file to *yymmddxx.mls*. In this instance, *xx* are sequential characters ranging from 00 to 99, and *yymmdd* represents the current year, month, and day.

You can use this option to prune old message log files to free up disk space. The latest output is always appended to the file specified by -o or -ot.

You cannot use this option with the -on option.

See also

[“-o option” on page 13](#)

[“-on option” on page 14](#)

[“-ot option” on page 16](#)

[“-v option” on page 21](#)

-ot option

Function	Logs output messages to the message log file, but truncates it first.
Syntax	dbmlsrv9 -c "connection-string" -ot logfilename ...
Description	Truncate the message log file and then append output messages to it. The default is to send output to the screen.
See also	“-o option” on page 13 “-on option” on page 14 “-os option” on page 15 “-v option” on page 21

-ps option

Function	Sets the maximum number of prepared statements to cache per connection.
Syntax	dbmlsrv9 -c "connection-string" -ps num ...
Description	Controls the maximum number of ODBC prepared statements kept in the prepared statement cache. Caching prepared statements improves performance, but consumes resources. Some consolidated database types have configurable limits on the number of prepared statements, so this option may be set accordingly.

-q option

Function	Instructs MobiLink to run in a minimized window on startup.
Syntax	dbmlsrv9 -c "connection-string" -q ...
Description	Minimize the MobiLink synchronization server window.

-r option

Function	Sets the maximum number of deadlock retries.
Syntax	dbmlsrv9 -c "connection-string" -r retries ...
Description	By default, MobiLink synchronization server retries uploads that are

deadlocked, for a maximum of 10 attempts. If the deadlock is not broken, synchronization fails, since there is no guarantee that the deadlock can be overcome. This option allows an arbitrary retry limit to be set. To stop the server from retrying deadlocked transactions, specify **-r 0**. The upper bound on this setting is 2 to the power 32, minus one.

-rd option

Function	Sets the maximum delay time between deadlock retries.
Syntax	dbmlsrv9 -c "connection-string" -rd delay ...
Description	When upload transactions are deadlocked, the MobiLink synchronization server waits a random length of time before retrying the transaction. The random nature of the delay increases the likelihood that future attempts will succeed. This option allows you to specify the maximum delay in units of seconds. The value 0 (zero) makes retries instantaneous, but larger values are recommended because they yield more successful retries. The default and maximum delay value is 30.

-s option

Function	Sets the maximum number of rows fetched, inserted, updated, or deleted at once.
Syntax	dbmlsrv9 -c "connection-string" -s count ...
Description	<p>Set the maximum number of rows transferred between the MobiLink synchronization server and the consolidated database to <i>count</i>.</p> <p>Set this option to no less than the number of rows specified in the ODBC prefetch option, if this option is set. The default value is 10.</p> <p>The number of rows fetched at once can be viewed in the log file as rowset size.</p> <p><i>Note:</i> The actual maximum number of rows transferred is influenced by settings in your ODBC data source, and/or database client software.</p>

-sl dnet option

Function	Sets the .NET Common Language Runtime (CLR) options and forces the CLR to load on startup.
Syntax	dbmlsrv9 -c "connection-string" -sl dnet options ...
Description	Sets options to pass directly to the .NET CLR. The options are:

Option	Description
-Dname=value	Set an environment variable. For example, <code>-Dsynchtype=far -Dextra_rows=yes</code> For more information, see the .NET framework class System.Environment.
-MLAutoLoadPath=path	Set the location of base assemblies. Only works with private assemblies. To tell MobiLink where assemblies are located, use this option or -MLDomConfigFile, but not both. When you use -MLAutoLoadPath, you cannot specify a domain in the event script. The default is the current directory.
-MLDomConfigFile=file	Set the location of base assemblies. Use when you have shared assemblies, or you don't want to load all assemblies in the directory, or you can't use MLAutoLoadPath for some other reason. To tell MobiLink where assemblies are located, use -MLDomConfigFile or -MLAutoLoadPath, but not both.
-MLStartClasses=classnames	At server startup, load and instantiate user-defined start classes in the order listed.
-clrConGC	Enable concurrent garbage collection in the CLR.
-clrFlavor=(wks svr)	Flavor of the .NET CLR to load. The flavor is svr for server and wks for workstation. By default, wks is loaded.
-clrVersion=version	Version of the .NET CLR to load. This must be prefixed with v . For example, v1.0.3705 loads the directory <code>\WINNT\Microsoft.NET\Framework\v1.0.3705</code> .

To display this list of options, use the following command:

```
dbmlsrv9 -sl dnet (?)
```

See also

“Writing Synchronization Scripts in .NET” [*MobiLink Synchronization User's Guide*, page 251]

-sl java option

Function	Sets the Java virtual machine options and forces the virtual machine to load on startup.
Syntax	dbmlsrv9 -c "connection-string" -sl java options ...
Description	Sets -jrepath and other options to pass directly to the Java virtual machine. The options are:

Option	Description
(-hotspot -server -classic)	Override the default choice for the Java VM to use.
{ -cp -classpath } <i>location</i> ; ...	Specify a set of directories or jar files in which to search for classes.
-Dname=value	Set a system property. For example, -Dsynchtype=far -Dextra_rows=yes
-DMLStartClasses=class, ...	At server startup, load and instantiate user-defined start classes in the order listed.
-jrepath path	Override the default JRE path, which is the <i>sun\jre131</i> directory under the <i>Sybase\shared</i> directory.
-verbose:(class gc jni)	Enable verbose output.
-X vm-option	Set a VM-specific option as described in the file <i>Xusage.txt</i> , which by default is installed to <i>Sybase\Shared\Sun\jre131\bin\hotspot</i> .

To display this list of options, use the following command:

```
dbmlsrv9 -sl java (?)
```

See also	“Writing Synchronization Scripts in Java” [<i>MobiLink Synchronization User’s Guide</i> , page 227]
----------	--

-t option

Function	Creates a file containing all the ODBC calls issued by MobiLink.
Syntax	dbmlsrv9 -c "connection-string" -t ODBC-output-file ...

Description	This option can be used to create a file containing all of the ODBC calls issued by MobiLink. If used on UNIX, with the Adaptive Server Anywhere driver used as a driver manager, this feature is ignored. The feature is useful for tracing what was called, passed, and retrieved. It has a severe impact on performance, so should not be used in production.
-------------	--

To prevent the file from becoming large, use the [“-tt option” on page 20](#).

See also	“-tt option” on page 20
----------	---

-tt option

Function	Logs ODBC calls issued by MobiLink to a file. If the file already exists, it first deletes it.
----------	--

Syntax	dbmlsrv9 -c "connection-string" -tt ODBC-output-file ...
--------	---

Description	This option is used to create a file containing all of the ODBC calls issued by MobiLink. If used on UNIX with the Adaptive Server Anywhere driver used as a driver manager, this feature is ignored. The feature is useful for tracing what was called, passed, and retrieved. It has a severe impact on performance, so should not be used in production.
-------------	---

See also	“-t option” on page 19
----------	--

-u option

Function	Sets the upload cache size.
----------	-----------------------------

Syntax	dbmlsrv9 -c "connection-string" -u size[k m g] ...
--------	---

Description	The amount of space, in bytes, to reserve for caching upload streams that are being processed. Use the suffix k, m, or g to specify units of kilobytes, megabytes, or gigabytes respectively. You should consider enlarging this value if your clients upload large streams, or many clients synchronize at once, or both. The suggested size is the maximum expected size of an upload stream multiplied by the number of worker threads. The default value is 500 Kb.
-------------	---

See also	“-bc option” on page 9
----------	--

-ud option

Function	Instructs MobiLink to run as a daemon.
----------	--

Syntax	dbmlsrv9 -c "connection-string" -ud ...
--------	--

Description	UNIX platforms only.
-------------	----------------------

See also “Running MobiLink Outside the Current Session” [*MobiLink Synchronization User’s Guide*, page 329]

-v option

Function Allows you to specify what information is logged to the message log file and displayed in the synchronization window.

Syntax **dbmlsrv9 -c "connection-string" -v[levels] ...**

Description This option controls the type of messages written to the message log file.

If you specify `-v` alone, the MobiLink synchronization server writes a minimal amount of information about each synchronization.

The values of *levels* are as follows. You can use one or more of these options at once; for example, `-vnrsu`.

- ◆ **+** Turn on all logging options that increase verbosity.
- ◆ **c** Show the content of each synchronization script when it is invoked. This level implies **s**.
- ◆ **f** Show first-read errors. This will log errors caused when load-balancing devices check for server liveness by making connections which don’t send any data, and thus result in failed synchronizations.
- ◆ **h** Show the remote schema as uploaded during synchronization.
- ◆ **n** Show row-count summaries.
- ◆ **p** Show progress offsets.
- ◆ **r** Display the column values of each row uploaded or downloaded.
- ◆ **s** Show the name of each synchronization script as it is invoked.
- ◆ **t** Show the translated SQL that results from scripts that are written in ODBC canonical format. This level implies **c**. The following example shows the automatic translation of a statement for Adaptive Server Anywhere.

```
I. 02/11 11:02:14. [102]: begin_upload synch2
    { call SynchLogLine( ?, ?, 'begin_upload' ) }
I. 02/11 11:02:14. [102]: Translated SQL:
    call SynchLogLine( ?, ?, 'begin_upload' )
```

The following example shows the translation of the same statement for Microsoft SQL Server.

```
I. 02/11 11:03:21. [102]: begin_upload synch2
    { call SynchLogLine( ?, ?, 'begin_upload' ) }
I. 02/11 11:03:21. [102]: Translated SQL:
    EXEC SynchLogLine ?, ?, 'begin_upload'
```

- ◆ **u** Show undefined table scripts. This may help new users understand the synchronization process.

-w option

Function Sets the number of worker threads.

Syntax **dbmlsrv9 -c "connection-string" -w count ...**

Description Each worker thread accepts synchronization requests one at a time. Each worker thread is associated with a network protocol. If you have more than one protocol defined, the worker threads are divided evenly among the protocols.

Each worker thread uses one connection to the consolidated database. The MobiLink synchronization server opens one additional connection for administrative purposes. Hence, the minimum number of connections from the MobiLink synchronization server to the consolidated database is *count* + 1.

The number of worker threads has a strong influence on MobiLink synchronization throughput, and you need to run tests to determine the optimum number for your particular synchronization setup. The number of worker threads determines how many synchronizations can be active simultaneously; the rest will be queued waiting for worker threads to become available. Thus adding worker threads should increase throughput, but it will also increase the possibility of contention between the active synchronizations. At some point adding more worker threads will decrease throughput, because the increased contention outweighs the benefit of overlapping synchronizations.

☞ For more information, see the MobiLink Performance whitepaper at <http://my.sybase.com/detail?id=1009664>, and “MobiLink Performance” [*MobiLink Synchronization User's Guide*, page 285].

The value set for this option is also the default setting for the **-wu** option, which can be used to limit the number of threads that can simultaneously upload. This is useful if the optimum number of worker threads for downloading is larger than the optimum number for uploading, as is typically the case with remote databases on slow computers or with slow connections to the MobiLink server. Tests have shown that for slow synchronization clients (such as Palm devices or computers connected by

dialup), the best throughput is achieved with a large number of worker threads (via `-w`) with a small number allowed to apply uploads simultaneously (via `-wu`). In general, the optimum number for `-wu` depends on the consolidated database, and is relatively independent of the processing or network speeds for the remote databases. Therefore, when you increase the number of threads with `-w`, you may want to use `-wu` to restrict the number that can upload simultaneously. For more information, see “[-wu option](#)” on page 23.

The default number of worker threads is 5.

-wu option

Function Sets the maximum number of worker threads that can apply uploads simultaneously.

Syntax **dbmlsrv9 -c "connection-string" -wu count ...**

Description Use the `-wu` option to limit the number of worker threads that can simultaneously apply uploads. When the limit is reached, a worker thread that is ready to apply its upload must wait until another finishes its upload. The excess worker threads are still free to receive uploads or to download.

The most common cause of contention in the consolidated database is having too many worker threads applying uploads simultaneously. This can be an issue when the network connection is slow, or when the client device has low processing speed. For example, when working over a wide-area wireless network or using a Palm device you may want to increase the total number of threads (`-w`) but limit the number that can apply uploads simultaneously.

Consider the following example. In a pilot setup using a LAN and remote databases on PCs, you find that the optimum number of worker threads is approximately 10 for both upload-only and download-only synchronizations, and that corresponds to 100% CPU utilization on the consolidated database. With fewer worker threads you find that throughput is less and the CPU utilization for the consolidated database is lower. With more worker threads, throughput does not increase because the consolidated database is already processing as fast as it can with 10 workers. When you switch to using a dialup network with 10 MobiLink worker threads, you will probably find that throughput is lower and the consolidated CPU utilization has dropped. You may find that you can get throughput (and consolidated CPU utilization) to approach the values obtained with the LAN by increasing the number of worker threads (via `-w`) while keeping the number that apply uploads simultaneously limited to 10 (via `-wu`).

By default, all worker threads can apply uploads simultaneously. The number of worker threads that are used is set by the `-w` option. The default

is 5.

-x option

Function	Sets communications protocol and parameters for MobiLink clients. These are used by the MobiLink synchronization server to listen for synchronization requests.
Syntax	dbmlsrv9 -c "connection-string" -x protocol[(network-parameters;...)]...
Description	Specify communications protocol through which to communicate with client applications. The default is TCPIP with port 2439.

Note for UltraLite users

If you are using an UltraLite Java application *and* you are using TLS security, the syntax of -x is slightly different. For details, see “Using transport-layer security” [*UltraLite Static Java User’s Guide*, page 48].

The allowed values of *protocol* are as follows:

- ◆ **tcpip** Accept connections from applications via TCP/IP.
- ◆ **http** Accept connections via the standard Web protocol. Client applications can pick their HTTP version and the MobiLink synchronization server adjusts on a per-connection basis.
- ◆ **https** Accept connections via a variant of HTTP that handles secure transactions. The HTTPS stream implements HTTP over SSL/TLS using RSA encryption, and is compatible with any other HTTPS server.

Optionally, you can also specify network parameters, in the form *parameter=value*. Separate multiple parameters with semicolons. Which parameters you specify depends on the protocol you choose.

- ◆ **TCP/IP parameters** If you specify the **tcpip** protocol, you can optionally specify the following *network-parameters*:
 - **client_port=nnnnn or client_port=nnnnn-mmmmm** A range of client ports for communication. If only one value is specified, the end of the range is 100 greater than the initial value, for a total of 101 ports. The option can be useful for clients inside a firewall communicating with a MobiLink synchronization server outside the firewall.
 - **host=hostname** The host name or IP number on which the MobiLink synchronization server should listen. The default value is **localhost**.

- **liveness_timeout=n** The amount of time, in seconds, after a client stops communicating before MobiLink recovers the connection. A value of 0 means that there is no timeout. This option is only effective if download acknowledgement is set to off (the default). The default is 120 seconds.
- **port=portnumber** The socket port number on which the MobiLink synchronization server should listen. The default port is 2439, which is the IANA registered port number for the MobiLink synchronization server.
- **security=cipher(keyword=value;...)** All communication through this connection is to be encrypted using the cipher suite specified. The cipher can be one of **ecc_tls** or **rsa_tls**. These refer to elliptic-curve and RSA certification. For backwards compatibility, **ecc_tls** can also be specified as **certicom_tls**.

You can optionally specify the security keywords **certificate** (the certificate that is to be used for server authentication), and **certificate_password**. Certificate value is a file. You must use a certificate that matches the cipher suite you choose. If you do not specify a certificate, MobiLink uses a sample certificate appropriate to the cipher suite you chose.

Your installation includes a sample elliptic-curve certificate called **sample.crt** with password **tJ1#m6+W**, and a sample RSA certificate called **rsaserver.crt** with password **test**. The sample certificates are for testing and development only. They provide no security because the same certificates and passwords are distributed to all Adaptive Server Anywhere customers. New certificates are available from several companies, including Entrust Technologies and VeriSign.

Separately licensable option required

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 about security, see “Transport-Layer Security” [*MobiLink Synchronization User’s Guide*, page 337].

- ♦ **HTTP parameters** If you specify the **http** protocol, you can optionally specify the following *network-parameters*:
 - **client_port=nnnnn or client_port=nnnnn-mmmmm** A range of client ports for communication. If only one value is specified, the end of the range is 100 greater than the initial value, for a total of 101 ports.
 - **contd_timeout=seconds** The number of seconds the MobiLink synchronization server waits to receive the next part of a partially

completed synchronization before the synchronization is abandoned. You can tune this option to free MobiLink worker threads when the wait time indicates that the client will never continue the connection. The default value is 30 seconds.

- **host=hostname** The host name or IP number on which the MobiLink synchronization server should listen. The default value is **localhost**.
- **port=portnumber** The socket port number on which the MobiLink synchronization server should listen. The port number must be a decimal number that matches the port the MobiLink synchronization server is setup to monitor. The default port is 80.
- **security=cipher(keyword=value;...)** All communication through this connection is to be encrypted using the cipher suite specified. The cipher can be one of **ecc_tls** or **rsa_tls**. These refer to elliptic-curve and RSA certification. For backwards compatibility, **ecc_tls** can also be specified as **certicom_tls**.

You can optionally specify the security keywords, **certificate** (the certificate that is to be used for server authentication), and **certificate_password**. Certificate value is a file. You must use a certificate that matches the cipher suite you choose. If you do not specify a certificate, MobiLink uses a sample certificate appropriate to the cipher suite you chose.

Your installation includes a sample elliptic-curve certificate called **sample.crt** with password **tJ1#m6+W**, and a sample RSA certificate called **rsaserver.crt** with password **test**. The sample certificates are for testing and development only. They provide no security because the same certificates and passwords are distributed to all Adaptive Server Anywhere customers. New certificates are available from several companies, including Entrust Technologies and VeriSign.

Separately licensable option required

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 about security, see “Transport-Layer Security” [*MobiLink Synchronization User’s Guide*, page 337].

- **unknown_timeout=seconds** The number of seconds the MobiLink synchronization server waits to receive HTTP headers on a new connection before the synchronization is abandoned. You can tune this option to free MobiLink worker threads when the wait time indicates that a network failure has occurred. The default value is 30 seconds.

- **url_suffix=suffix** The suffix to add to the URL on the first line of each HTTP request. This parameter can be used to help ensure that a particular client connects to the intended server. Values must match or synchronization will not be successful.
- **version=http-version** The MobiLink synchronization server automatically detects the HTTP version used by a client. This parameter is a string specifying the default version of HTTP to use in case the server cannot detect the method used by the client. You have a choice of **1.0** or **1.1**. The default value is **1.1**.
- ◆ **HTTPS parameters** The HTTPS communication stream uses Certicom RSA security.

Separately licensable option required


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 about security, see “Transport-Layer Security” [*MobiLink Synchronization User’s Guide*, page 337].

If you specify the **https** protocol, you can optionally specify the following *network-parameters*:

- **client_port=nnnnn or client_port=nnnnn-mmmmm** A range of client ports for communication. If only one value is specified, the end of the range is 100 greater than the initial value, for a total of 101 ports.
- **contd_timeout=seconds** The number of seconds the MobiLink synchronization server waits to receive the next part of a partially completed synchronization before the synchronization is abandoned. You can tune this option to free MobiLink worker threads when the wait time indicates that the client will never continue the connection. The default value is 30 seconds.
- **host=hostname** The host name or IP number on which the MobiLink synchronization server should listen. The default value is **localhost**.
- **port=portnumber** The socket port number on which the MobiLink synchronization server should listen. The port number must be a decimal number that matches the port the MobiLink synchronization server is setup to monitor. The default port is 443.
- **certificate** An optional parameter that specifies a certificate name. This must be an RSA certificate. If you do not specify a certificate, MobiLink uses the sample RSA certificate that is provided with Adaptive Server Anywhere. It is called *rsaserver.crt* and has the password test.

The sample certificates are for testing and development only. They provide no security because the same certificates and passwords are distributed to all Adaptive Server Anywhere customers. New certificates are available from several companies, including Entrust Technologies and VeriSign.

- **certificate_password** An optional parameter that specifies a password for the certificate.
 For more information about security, see “Transport-Layer Security” [*MobiLink Synchronization User’s Guide*, page 337].
- **unknown_timeout=seconds** The number of seconds the MobiLink synchronization server waits to receive HTTP headers on a new connection before the synchronization is abandoned. You can tune this option to free MobiLink worker threads when the wait time indicates that a network failure has occurred. The default value is 30 seconds.
- **url_suffix=suffix** The suffix to add to the URL on the first line of each HTTP request. This parameter can be used to help ensure that a particular client connects to the intended server. Values must match or synchronization will not be successful.
- **version=http-version** The MobiLink synchronization server automatically detects the HTTP version used by a client. This parameter is a string specifying the default version of HTTP to use in case the server cannot detect the method used by the client. You have a choice of **1.0** or **1.1**. The default value is **1.1**.

-za option

Function	Generates statement-based scripts that perform a simple snapshot synchronization.
Syntax	dbmlsrv9 -c "connection-string" -za
Description	<p>The following scripts are generated:</p> <ul style="list-style-type: none">◆ upload_insert◆ upload_update◆ upload_delete◆ download_cursor <p>This option generates active scripts; that is, they are used for the current synchronization. The scripts are also saved and will work for subsequent synchronizations using the same script version. The -za option is typically used for quick demos. To generate scripts as a starting point for writing your own scripts, the dbmlsrv9 -ze option might be more useful.</p>

To generate scripts, you must also specify that the client sends column names. You do this when you initiate synchronization. For Adaptive Server Anywhere remotes, see [“SendColumnNames \(scn\) extended option” on page 62](#). For UltraLite remotes, see [“Send Column Names synchronization parameter”](#) [*UltraLite Database User’s Guide*, page 171].

The generated scripts perform one-to-one snapshot synchronization using the table and column names sent from the client. If the consolidated database has different table or column names than the remote, activating these scripts will cause an error during the synchronization.

Note:

Scripts are generated the first time that a remote synchronizes with a script version that doesn’t exist. If the given script version already exists, -za has no effect. This means that you cannot use -za to generate scripts one table at a time for the same script version. Using -za, you must generate scripts for all tables and publications at once.

See also [“-ze option” on page 29](#)

Example The following dbmlsrv9 command enables automatic script generation. The dbmlsync command sets the necessary SendColumnNames option.

```
dbmlsrv9 -c "dsn=YourDBDSN" -za
dbmlsync -c dsn=dsn_remote -e "SendColumnNames=ON"
```

-ze option

Function Generates sample scripts that, if activated, perform a simple snapshot synchronization.

Syntax **dbmlsrv9 -c "connection-string" -ze**

Description The following scripts are generated:

- ◆ example_upload_insert
- ◆ example_upload_update
- ◆ example_upload_delete
- ◆ example_download_cursor

You can use these scripts as starting points for creating your own scripts.

To generate scripts, you must also specify that the client sends column names. You do this when you initiate synchronization. For Adaptive Server Anywhere remotes, see [“SendColumnNames \(scn\) extended option” on](#)

[page 62](#). For UltraLite remotes, see “Send Column Names synchronization parameter” [*UltraLite Database User’s Guide*, page 171].

The generated scripts perform one-to-one snapshot synchronization using the table and column names sent from the client. If the consolidated database has different table or column names than the remote, activating these scripts will cause an error during the synchronization.

Note:

Scripts are generated only the first time that a remote synchronizes, and only when the given script version does not exist. Otherwise, -ze has no effect.

See also [“-za option” on page 28](#)

-zp option

Function Adjusts which timestamp values will be used for conflict detection purposes.

Syntax **dbmlsrv9 -c "connection-string" -zp**

Description 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. The option is useful when timestamps in the consolidated database are more precise than in the remote, as updated timestamps on the remote can cause conflicts in the next synchronization. The option allows MobiLink to ignore these conflicts. When there is a precision mismatch and -zp is not used, a per synchronization and a schema sensitive per table warning are written to the log to advertise the -zp option. Another per synchronization warning is also added to advise users to adjust the timestamp precision on the remote database where possible.

-zs option

Function Specifies a MobiLink server name for dbmlstop purposes.

Syntax **dbmlsrv9 -c "connection-string" -zs name**

Description Specify a server name for the MobiLink synchronization server. If the MobiLink synchronization server is started using the -zs option, it must be shut down using the dbmlstop server-name command. Shutdown may only be initiated from the computer where the MobiLink synchronization server is installed.

See also [“MobiLink stop utility” on page 303](#)

-zt option

Function	Specifies the maximum number of processors used to run the MobiLink synchronization server.
Syntax	dbmlsrv9 -c "connection-string" -zt number
Description	<p>This option may be required for some ODBC drivers. It also gives you fine control of processor resources.</p> <p>This option can only be used on Windows operating systems. The default is the number of processors on the computer.</p>

-zu option

Function	Controls the automatic addition of users when the authenticate_user script is undefined.
Syntax	dbmlsrv9 -c "connection-string" -zu{ + - } ...
Description	If this is supplied as -zu+, then unrecognized MobiLink user names are added to the ml_user table on first synchronizing. If the argument is supplied as -zu-, or not supplied, unrecognized user names are prevented from synchronizing.

-zw option

Function	Controls which levels of warning message to display.
Syntax	dbmlsrv9 -c "connection-string" -zw levels
Description	MobiLink has five levels of warning messages:

Level	Description
0	Suppress all warning messages
1	Server and high ODBC level: warning messages when the MobiLink synchronization server starts
2	Synchronization and user level: warning messages when a synchronization starts
3	Schema level: warning messages when a MobiLink synchronization server is processing a client schema
4	Script and lower ODBC level: warning messages when a MobiLink synchronization server fetches, prepares, or executes scripts
5	Table or row level: warning messages when a MobiLink synchronization server performs table operations in an upload or download

To specify the level of warning messages you want reported, you can separate levels with a comma, or separate a range with two dots. For example, **-zw 1..3,5** is the same as **-zw 1,2,3,5**.

The reporting of messages has a slight impact on performance. Levels with a higher number tend to produce more messages.

If **-zw** is used more than once in the same command line, MobiLink recognizes only the last instance. If settings of **-zw**, **-zwd**, and **-zwe** conflict, MobiLink gives priority to **-zwe**, then **-zwd**, then **-zw**.

The default is **1,2,3,4,5**, which indicates that all levels of warning message should be displayed.

-zwd option

Function	Disables specific warning codes.
Syntax	dbmlsrv9 -c "connection-string" -zwd code,...
Description	You can disable specific warning codes so that they will not be reported, even though other codes of the same level are reported.

☞ For a complete list of warning message codes, see [“MobiLink Synchronization Server Warning Messages” on page 437](#).

If `-zwd` is used more than once in the same command line, MobiLink accumulates the settings. If settings of `-zw`, `-zwd`, and `-zwe` conflict, MobiLink gives priority to `-zwe`, then `-zwd`, then `-zw`.

-zwe option

Function Enables specific warning codes.

Syntax **dbmlsrv9 -c "connection-string" -zwe code,...**

Description You can enable specific warning codes so that they will be reported even though you have disabled other codes of the same level using `-zw`.

☞ For a complete list of warning message codes, see [“MobiLink Synchronization Server Warning Messages” on page 437](#).


If `-zwe` is used more than once on the same command line, MobiLink accumulates the settings. If settings of `-zw`, `-zwd`, and `-zwe` conflict, MobiLink gives priority to `-zwe`, then `-zwd`, then `-zw`.

CHAPTER 2

MobiLink Synchronization Client

About this chapter

This chapter describes details of the MobiLink synchronization client, dbmlsync. It is used to synchronize Adaptive Server Anywhere remote databases with a consolidated database.

 The dbmlsync utility only works with Adaptive Server Anywhere remote databases. To synchronize UltraLite remote databases, see “UltraLite Clients” [*MobiLink Synchronization User’s Guide*, page 207].

Contents

Topic:	page
MobiLink synchronization client	36
dbmlsync options	40

MobiLink synchronization client

Use the dbmlsync utility to synchronize Adaptive Server Anywhere remote databases with a consolidated database.

Syntax

dbmlsync [*options*] [*transaction-logs-directory*]

Option	Description
-a	Do not prompt for input again on error. See “ -a option ” on page 40.
-ap	Specify authentication parameters. See “ -ap option ” on page 40.
-ba <i>filename</i>	Apply a download file. See “ -ba option ” on page 40.
-bc <i>filename</i>	Create a download file. See “ -bc option ” on page 40.
-be <i>string</i>	When creating a download file, add a string. See “ -be option ” on page 41.
-bg	When creating a download file, make it suitable for new remotes. See “ -bg option ” on page 41.
-c <i>connection-string</i>	Supply database connection parameters in the form <i>parm1=value1; parm2=value2,...</i> If you do not supply this option, a dialog will appear and you must supply connection information. See “ -c option ” on page 42.
-d	Drop any other connections to the database whose locks conflict with the articles to be synchronized. See “ -d option ” on page 42.
-dl	Display log messages on the console. See “ -dl option ” on page 43.
-ds	Specify download-only synchronization. See “ -ds option ” on page 43.
-e “ <i>option=value</i> ”...	Specify extended options. See “ -e extended options ” on page 44.
-eh	Ignore errors that occur in hook functions.
-ek <i>key</i>	Specify encryption key. See “ -ek option ” on page 71.
-ep	Prompt for encryption key. See “ -ep option ” on page 71.

Option	Description
-eu	Specify extended options for upload defined by most recent -n option. See “ -eu option ” on page 71.
-i filename	Execute file containing SQL statements immediately after synchronization. See “ -i option ” on page 72.
-is	Ignore schedule. See “ -is option ” on page 72.
-k	Close window on completion. See “ -k option ” on page 72.
-l	List available extended options. See “ -l option ” on page 72.
-mn password	Specify new MobiLink password. See “ -mn option ” on page 73.
-mp password	Specify MobiLink password. See “ -mp option ” on page 73.
-n name	Specify synchronization publication name(s). See “ -n option ” on page 73.
-o logfile	Log output messages to this file. See “ -o option ” on page 74.
-os size	Maximum size of output file. See “ -os option ” on page 74.
-ot logfile	Truncate file and log output messages to file. See “ -ot option ” on page 75.
-p	Disable logscan polling. See “ -p option ” on page 75.
-pd dllname;...	Preload specified dlls for Windows CE. See “ -pd option ” on page 75.
-pi	Test that you can connect to MobiLink. See “ -pi option ” on page 76.
-pp number	Set logscan polling period. See “ -pp option ” on page 77.
-q	Run in minimized window. See “ -q option ” on page 77.
-r[a b]	Upload retry on client progress. See “ -r option ” on page 77.

Option	Description
-sc	Reload schema information before each synchronization. See “-sc option” on page 79.
-u <i>ml_username</i>	Allows you to specify the MobiLink user to synchronize. See “-sc option” on page 79.
-uo	Synchronization will be upload-only (no download). See “-uo option” on page 80.
-urc <i>row-estimate</i>	Allows you to specify an estimate of the rows that will be uploaded. See “-urc option” on page 80.
-v [<i>levels</i>]	Verbose operation. See “-v option” on page 80.
-wc <i>classname</i>	Specify a Windows class name for ActiveSync synchronization (Windows CE only). See “-wc option” on page 81.
-x	Rename and restart the transaction log. See “-x option” on page 82.
<i>transaction-logs-directory</i>	Specify the location of the transaction log. See Transaction Log File, below.

Description

Run dbmlsync on the command line to synchronize an Adaptive Server Anywhere remote database with a consolidated database.

To locate and connect to the MobiLink synchronization server, dbmlsync uses the information on the publication, synchronization user, synchronization subscription, or command line.

Transaction log file The *transaction-logs-directory* is the directory that contains the transaction log for the Adaptive Server Anywhere remote database. There is an active transaction log and transaction log archive files, both of which may be required by dbmlsync to determine what to upload. You must specify this parameter if the following are true:

- ◆ the working log file has been truncated and renamed since you last synchronized
- ◆ you run the dbmlsync utility from a directory other than the one where the renamed log files are stored

☞ For more information, see “Transaction log files” [*MobiLink Synchronization User’s Guide*, page 187].

Using a configuration file You can put dbmlsync command line options in a configuration file and optionally use the File Hiding utility, dbfhide, to add

simple encryption to the configuration file. For more information, see “Using configuration files” [ASA Database Administration Guide, page 10] and “The File Hiding utility” [ASA Database Administration Guide, page 466].

dbmlsync event hooks There are also dbmlsync client stored procedures that can help you customize the synchronization process. For more information, see “Customizing the client synchronization process” [MobiLink Synchronization User’s Guide, page 194] and [“Client event-hook procedures” on page 269](#).

Using dbmlsync For more information about using dbmlsync, see “Initiating synchronization” [MobiLink Synchronization User’s Guide, page 185].


dbmlsync options

This section lists MobiLink synchronization client command line options.


-a option

Function	Specifies that dbmlsync should not prompt for input again on error.
Syntax	dbmlsync -a ...

-ap option

Function	Specifies parameters for authentication.
Syntax	dbmlsync -ap "parameters,..." ...
Description	Use when you use the <code>authenticate_parameters</code> connection script. For example, <pre>dbmlsync -ap "parm1,parm2,parm3"</pre>  For more information, see “authenticate_parameters connection event” on page 98 .

-ba option

Function	Applies a download file.
Syntax	dbmlsync -ba "filename" ...
Description	Specify the name of an existing download file. You can optionally specify a path. If you do not specify a path, the default location is dbmlsync’s current working directory (the directory where dbmlsync was started).  For more information, see “File-Based Downloads” [<i>MobiLink Synchronization User’s Guide</i> , page 117].

-bc option

Function	Creates a download file.
Syntax	dbmlsync -bc "filename" ...
Description	Create a download file with the specified name. You should use the file extension <code>.df</code> for download files. You can optionally specify a path. If you do not specify a path, the default location is dbmlsync’s current working directory, which is the directory

where dbmlsync was started.

Optionally, in the same dbmlsync command line as you create the download file, you can use the `-be` option to specify a string that can be validated at the remote database, and the `-bg` option to create a download file for new remote databases.

See also [“File-Based Downloads” \[MobiLink Synchronization User’s Guide, page 117\]](#)
 [“-be option” on page 41](#)
 [“-bg option” on page 41](#)

-be option

Function	When creating a download file, this option specifies an extra string to be included in the file.
Syntax	dbmlsync -bc "filename" -be "string" ...
Description	The string can be used for authentication or other purposes. It is verified at the remote database using the <code>sp_hook_dbmlsync_validate_download_file</code> stored procedure.
See also	“sp_hook_dbmlsync_validate_download_file” on page 295 “File-Based Downloads” [MobiLink Synchronization User’s Guide, page 117] “-bc option” on page 40

-bg option

Function	When creating a download file, this option creates a file that can be used with remote databases that have not yet synchronized.
Syntax	dbmlsync -bc "filename" -bg ...
Description	<p>The <code>-bg</code> option causes the download file to update the generation numbers on the remote database.</p> <p>This option allows you to build a download file that can be applied to remote databases that have never synchronized. Otherwise, you must perform a synchronization before you apply a download file.</p> <p>Download files built with the <code>-bg</code> option should be snapshot downloads. Timestamp-based downloads will not work with remote databases that have not synchronized because the last download timestamp on a new remote is by default January 1, 1900, which will be earlier than the last download timestamp in the download file. For timestamp-based file-based downloads</p>

to work, the last download timestamp in the download file must be the same or earlier than on the remote.

You should not apply download files built with the `-bg` option to remote databases that have already synchronized. The `-bg` option causes the generation numbers on the remote database to be updated with the value on the consolidated database at the time the download file was created. For remotes that have already synchronized, this means that the remote database is not forced to upload data before applying a download, and so data could be lost.

See also

[“-bc option” on page 40](#)

“File-Based Downloads” [*MobiLink Synchronization User’s Guide*, page 117]

“MobiLink generation numbers” [*MobiLink Synchronization User’s Guide*, page 125]

“Synchronizing new remotes” [*MobiLink Synchronization User’s Guide*, page 120]

-c option

Function

Specifies connection parameters for the remote database.

Syntax

dbmlsync -c "*connection-string*" ...

Description

The connection string must give dbmlsync permission to connect to the Adaptive Server Anywhere remote database. Commonly, a user ID with REMOTE DBA authority is used.

Specify the connection string in the form *keyword=value*, with multiple parameters separated by semicolons. If any of the parameter names contain spaces, you need to enclose the connection string in double quotes.

If you do not specify `-c`, a dbmlsync Setup dialog appears. You can specify the remaining command line options in the fields of the connection dialog.

For a complete list of connection parameters for connecting to Adaptive Server Anywhere databases, see “Connection parameters” [*ASA Database Administration Guide*, page 174].

-d option

Function

Drops conflicting locks to the remote database.

Syntax

dbmlsync -d ...

Description

During synchronization, unless the locktables extended option is set to OFF,

all tables involved in the publications being synchronized are locked to prevent any other processes from making changes. Ordinarily, if another process has a lock on one of these tables, the synchronization is delayed until that process releases its lock. Specifying this option forces Adaptive Server Anywhere to drop any other connections to the remote database that hold conflicting locks.

-dl option

Function	Displays messages in the log file.
Syntax	dbmlsync -dl . . .
Description	Normally when output is logged to a file, more messages are written to the log file than to the dbmlsync window. This option forces dbmlsync to write information normally only written to the file to the window as well. Using this option may have an effect on the speed of synchronization.

-ds option

Function	Specifies download-only synchronization.
Syntax	dbmlsync -ds . . .
Description	<p>When download-only synchronization occurs, dbmlsync does not upload any row operations or data. However, it does upload information about the schema and progress offset.</p> <p>In addition, dbmlsync ensures that changes on the remote are not overwritten during download-only synchronization. It does this by scanning the log to detect rows with operations waiting to be uploaded. If any of these rows is modified by the download stream, the download stream is rolled back and the synchronization fails. If the synchronization fails for this reason, you must do a full synchronization to correct the problem.</p> <p>When you have remotes that are synchronized by download-only synchronization, you should regularly do a full synchronization to reduce the amount of log that is scanned by the download-only synchronization. Otherwise, the download-only synchronizations will take an increasingly long time to complete.</p> <p>When -ds is used, the ConflictRetries setting is ignored. dbmlsync never retries a download-only synchronization. If a download-only synchronization fails, it will continue to fail until a normal synchronization is performed.</p>

See also [“DownloadOnly \(ds\) extended option” on page 49](#)

-e extended options

Function	Specifies extended options.
Syntax	dbmlsync -e <i>extended-option=value</i> ; ... <i>extended-option</i> : adr ctp cr p dbs el ft eh isc inc lt mem mp mn dir pp sch sv scn sa st sn to v vs vm vo vn vr vu
Parameters	Extended options can be specified by their long form or short form. See each option, below, for details.
Description	Options specified on the command line with the -e option apply to all synchronizations requested on the command line. For example, in the following command line, the extended option sv=test applies to the synchronization of both pub1 and pub2.

```
dbmlsync -e sv=test -n pub1 -n pub2
```

To specify extended options for a single upload, use the -eu option.

Extended options can be specified on the dbmlsync command line using the -e or -eu options, or they can be stored in the database. You store extended options in the database using Sybase Central, or by using the **OPTIONS** clause in any of the following statements:

- ◆ **CREATE SYNCHRONIZATION SUBSCRIPTION**
- ◆ **ALTER SYNCHRONIZATION SUBSCRIPTION**
- ◆ **CREATE SYNCHRONIZATION USER**
- ◆ **ALTER SYNCHRONIZATION USER**
- ◆ **CREATE SYNCHRONIZATION SUBSCRIPTION** without specifying a synchronization user (which associates extended options with a publication)

Dbmlsync combines options stored in the database with those specified on the command line. If conflicting options are specified, dbmlsync resolves them as follows. In the following list, options specified by methods occurring earlier in the list take precedence over those occurring later in the list.

- ◆ options specified on the command line with the -eu option
- ◆ options specified on the command line with the -e option

- ◆ options specified for the subscription (whether by a SQL statement or in Sybase Central)
- ◆ options specified for the user (whether by a SQL statement or in Sybase Central)
- ◆ options specified for the publication (whether by a SQL statement or in Sybase Central)

You can review extended options in the log and the SYSSYNC system table.

☞ For information on how extended options can be used to tune synchronization, see “Customizing synchronization” [*MobiLink Synchronization User’s Guide*, page 186].

For a detailed explanation of each option, see below.

See also [“-eu option” on page 71](#)

“SYSSYNC system table” [*ASA SQL Reference*, page 676]

Example The following dbmlsync command line illustrates how you can set extended options when you start dbmlsync:

```
dbmlsync -e "adr=host=localhost;dir=c:\db\logs"...
```

The following SQL statement illustrates how you can store extended options in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION TO mypub
FOR mluser
ADDRESS 'host=localhost'
OPTION schedule='weekday@11:30am-12:30pm', dir='c:\db\logs'
```

The following dbmlsync command line opens the usage screen that lists options and their syntax:

```
dbmlsync -l
```

CommunicationAddress (adr) extended option

Function Specifies the communication address for connecting to the MobiLink server.

Syntax **dbmlsync -e adr=network-parameters; ...**

Description For a list of *network-parameters*, see “CREATE SYNCHRONIZATION USER statement” [*MobiLink*] [*ASA SQL Reference*, page 351].

You must ensure that all subscriptions for a MobiLink user are synchronized to only one consolidated database. Otherwise, you may experience data loss and unpredictable behavior.

This option has a short form and long form: you can use **adr** or **CommunicationAddress**.

This option can also be stored in the database using the SQL statement that creates or alters a publication, subscription, or user. For more information, see “CREATE SYNCHRONIZATION USER statement [MobiLink]” [ASA *SQL Reference*, page 351].

Example

The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "adr=host=localhost"
```

To specify multiple network parameters on the command line, enclose them in single quotes. For example,

```
dbmlsync -e "adr='host=somehost;port=5001'"
```

To store the Address or CommunicationType in the database, you can use an extended option or you can use the ADDRESS or TYPE clause. For example,

```
CREATE SYNCHRONIZATION SUBSCRIPTION  
  TO sales_publication  
  FOR ml_user1  
  TYPE 'https'  
  ADDRESS host='localhost'
```

CommunicationType (ctp) extended option

Function	Specifies the communication type for connecting to the MobiLink server.
Syntax	dbmlsync -e ctp=sync-type; ...
Description	<p><i>sync-type</i> can be one of tcPIP, http, https, or ActiveSync. If you purchase a special license, you can also use ecc_tls and rsa_tls. The default is tcPIP.</p> <p>You must ensure that all subscriptions for a MobiLink user are synchronized to only one consolidated database. Otherwise, you may experience data loss and unpredictable behavior.</p> <p>This option has a short form and long form: you can use ctp or CommunicationAddress.</p> <p>This option can also be stored in the database using the SQL statement that creates or alters a publication, subscription, or user. For more information, see “CREATE SYNCHRONIZATION USER statement [MobiLink]” [ASA <i>SQL Reference</i>, page 351].</p>
See also	“Transport-Layer Security” [MobiLink Synchronization User’s Guide, page 337]

Example The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "ctp=https"
```

To store the Address or CommunicationType in the database, you can use an extended option or you can use the ADDRESS or TYPE clause. For example,

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  TYPE 'tcpip'
  ADDRESS host='localhost'
```

ConflictRetries (cr) extended option

Function Specifies the number of retries if the download fails because of conflicts.

Syntax **dbmlsync -e cr=number;** ...

Description -1 specifies that retries should continue indefinitely. The default is **-1**.

This option is useful only if the LockTables option is OFF, which is not the default.

This option has a short form and long form: you can use **cr** or **ConflictRetries**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

Example The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "cr=5"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION cr='5';
```

DisablePolling (p) extended option

Function Disables automatic logscan polling.

Syntax **dbmlsync -e p= { ON | OFF };** ...

Description In order to build an upload stream, dbmlsync must scan the transaction log. Usually it does this just before synchronization. However, when synchronizations are scheduled, dbmlsync by default scans the log in the time between scheduled synchronizations; and when the `sp_hook_dbmlsync_delay` hook is used, dbmlsync by default scans the log in the pause that occurs just before synchronization. This behavior is more efficient because the log is already at least partially scanned when synchronization begins. This default behavior is called logscan polling.

Logscan polling is on by default but only has an effect when synchronizations are scheduled or when `sp_hook_dbmlsync_delay` hook is used. When in effect, polling occurs at set intervals: dbmlsync scans to the end of the log, waits for the polling period, and then scans any new transactions in the log. By default, the polling period is 1 minute, but it can be changed with the dbmlsync `-pp` option or the `PollingPeriod` extended option.

The default is to not disable logscan polling (**OFF**).

This option is identical to **dbmlsync -p**.

This option has a short form and long form: you can use **p** or **DisablePolling**.

You can also store extended options in the database. For more information about dbmlsync extended options, see “[-e extended options](#)” on page 44.

See also “[PollingPeriod \(pp\) extended option](#)” on page 58

“[-p option](#)” on page 75

“[-pp option](#)” on page 77

Example The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "p=on"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION  
  TO sales_publication  
  FOR ml_user1  
  OPTION p='on';
```

DownloadBufferSize (dbs) extended option

Function Specifies the size of the download buffer.

Syntax **dbmlsync -e dbs=** *number* [**K** | **M**]; ...

Description	<p>The buffer size is specified in units of bytes. Use the suffix k or m to specify units of kilobytes or megabytes, respectively.</p> <p>If you set this option to 0, dbmlsync does not buffer the download stream. If the setting is greater than 0 but less than 4k, dbmlsync uses a 4k buffer size and issues a warning. The default is 32K on Windows CE, and 1M on all other operating systems.</p> <p>Download buffering increases the benefit of eliminating the download acknowledgement because it allows the worker thread to send the download faster.</p> <p>This option has a short form and long form: you can use dbs or DownloadBufferSize.</p> <p>You can also store extended options in the database. For more information about dbmlsync extended options, see “-e extended options” on page 44.</p>
Example	<p>The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:</p> <pre>dbmlsync -e "dbs=32k"</pre> <p>The following SQL statement illustrates how you can store this option in the database:</p> <pre>CREATE SYNCHRONIZATION SUBSCRIPTION TO sales_publication FOR ml_user1 OPTION dbs='32k' ;</pre>

DownloadOnly (ds) extended option

Function	Specifies that synchronization should be download-only.
Syntax	dbmlsync -e ds={ ON OFF }; ...
Description	<p>When download-only synchronization occurs, dbmlsync does not upload any row operations or data. However, it does upload information about the schema and progress offset.</p> <p>In addition, dbmlsync ensures that changes on the remote are not overwritten during download-only synchronization. It does this by scanning the log to detect rows with operations waiting to be uploaded. If any of these rows is modified by the download stream, the download stream is rolled back and the synchronization fails. If the synchronization fails for this reason, you must do a full synchronization to correct the problem.</p> <p>When you have remotes that are synchronized by download-only synchronization, you should regularly do a full synchronization to reduce the</p>

amount of log that is scanned by the download-only synchronization. Otherwise, the download-only synchronizations will take an increasingly long time to complete.

The default is **OFF** (full synchronization of both upload and download).

This option has a short form and long form: you can use **ds** or **DownloadOnly**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

See also

[“-ds option” on page 43](#)

Example

The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "ds=on"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION  
  TO sales_publication  
  FOR ml_user1  
  OPTION ds='ON';
```

ErrorLogSendLimit (el) extended option

Function

Specifies how much of the remote log file dbmlsync should send to the server when synchronization error occurs.

Syntax

dbmlsync -e el=number[K | M]; ...

Description

This option is specified in units of bytes. Use the suffix k or m to specify units of kilobytes or megabytes, respectively.

This option specifies the number of bytes of the output log that dbmlsync sends to the MobiLink synchronization server when errors occur during synchronization. Set this option to **0** if you don't want any dbmlsync output log to be sent.

If ErrorLogSendLimit is set to be large enough, dbmlsync sends the entire output log messages from the current session to the MobiLink synchronization server. For example, if the output log messages were appended to an old output log file, dbmlsync only sends the new messages generated in the current session. If the total length of new messages is greater than ErrorLogSendLimit, dbmlsync only logs the last part of the newly generated error and log messages up to the specified size.

Note: The size of the output log is influenced by your verbosity settings. You can adjust these using the `dbmlsync -v` option, or by using `dbmlsync` extended options starting with “verbose”. For more information, see “[-v option](#)” on page 80 and `-e` verbose options, below.

The default is **32K**.

This option has a short form and long form: you can use **el** or **ErrorLogSendLimit**.

You can also store extended options in the database. For more information about `dbmlsync` extended options, see “[-e extended options](#)” on page 44.

Example

The following `dbmlsync` command line illustrates how you can set this option when you start `dbmlsync`:

```
dbmlsync -e "el=32k"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION el='32k';
```

FireTriggers (ft) extended option

Function Specifies that triggers should be fired on the remote database when the download is applied.

Syntax **dbmlsync -e ft= { ON | OFF }; ...**

Description The default is **ON**.

This option has a short form and long form: you can use **ft** or **FireTriggers**.

You can also store extended options in the database. For more information about `dbmlsync` extended options, see “[-e extended options](#)” on page 44.

Example

The following `dbmlsync` command line illustrates how you can set this option when you start `dbmlsync`:

```
dbmlsync -e "ft=off"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION ft='off';
```

HoverRescanThreshold (hrt) extended option

Function	When you are using scheduling, this limits the amount of discarded memory that is allowed to accumulate before a rescan is performed.
Syntax	dbmlsync -e hrt= <i>number</i> [K M] ; ...
Description	<p>Specifies memory in units of bytes. Use the suffix k or m to specify units of kilobytes or megabytes, respectively. The default is 1M.</p> <p>When scheduling options are specified or when more than one dbmlsync -n option is specified, dbmlsync goes into hovering mode. Hovering is a feature that reduces the amount of time spent scanning the log when dbmlsync is started and asked to perform more than one synchronization before shutting down. Hovering can occur only when all the subscriptions to be synchronized involve the same MobiLink user.</p> <p>While hovering, dbmlsync keeps track of operations read from the log using a system that maintains information first in memory, and then spills it on to disk. As hovering continues, dbmlsync discards memory that has become fragmented. The amount of memory discarded is proportional to the number of operations processed while hovering and the size of the rows involved (not counting blobs). Memory is not discarded if the remote database has only one publication for the user being synchronized.</p> <p>Discarded memory can be recovered after a complete rescan is performed. There are two ways that you can control when memory is recovered: the HoverRescanThreshold extended option and the sp_hook_dbmlsync_log_rescan stored procedure.</p> <p>This option has a short form and long form: you can use hrt or HoverRescanThreshold.</p> <p>You can also store extended options in the database. For more information about dbmlsync extended options, see “-e extended options” on page 44.</p>
See also	“sp_hook_dbmlsync_log_rescan” on page 286
Example	<p>The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:</p> <pre>dbmlsync -e "hrt=2m"</pre> <p>The following SQL statement illustrates how you can store this option in the database:</p>

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION hrt='2m';
```

IgnoreHookErrors (eh) extended option

Function	Specifies that errors that occur in hook functions should be ignored.
Syntax	dbmlsync -e eh= { ON OFF }; ...
Description	<p>The default is OFF.</p> <p>This option has a short form and long form: you can use eh or IgnoreHookErrors.</p> <p>This option is equivalent to the dbmlsync -eh option.</p> <p>You can also store extended options in the database. For more information about dbmlsync extended options, see “-e extended options” on page 44.</p>
Example	<p>The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:</p> <pre>dbmlsync -e "eh=off"</pre> <p>The following SQL statement illustrates how you can store this option in the database:</p>

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION eh='off';
```

IgnoreScheduling (isc) extended option

Function	Specifies that scheduling settings should be ignored.
Syntax	dbmlsync -e isc= { ON OFF }; ...
Description	<p>If set to ON, dbmlsync ignores any scheduling information that is specified in extended options and synchronizes immediately. The default is OFF.</p> <p>This option is equivalent to the dbmlsync -is option.</p> <p>This option has a short form and long form: you can use isc or IgnoreScheduling.</p> <p>You can also store extended options in the database. For more information about dbmlsync extended options, see “-e extended options” on page 44.</p>

See also

“Scheduling synchronization” [*MobiLink Synchronization User’s Guide*, page 198]

[“Schedule \(sch\) extended option” on page 59](#)

Example

The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "isc=off"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION  
TO sales_publication  
FOR ml_user1  
OPTION isc='off';
```

Increment (inc) extended option

Function

Controls the size of incremental uploads.

Syntax

dbmlsync -e inc= *number* [**K** | **M**]; ...

Description

This option specifies a minimum incremental scan volume in units of bytes. Use the suffix k or m to specify units of kilobytes or megabytes, respectively.

When this option is specified, uploads are sent to MobiLink in one or more parts. This could be useful if a site has difficulty maintaining a connection for long enough to complete the full upload. When the option is not set, uploads are sent as a single unit.

The value of this option specifies, very approximately, the size of each upload part. The value of the option controls the size of each upload part as follows. Dbmlsync builds the upload stream by scanning the database transaction log. When this option is set, dbmlsync scans the number of bytes that are set in the option, and then continues scanning to the first point at which there are no outstanding partial transactions—the next point at which all transactions have either been committed or rolled back. It then sends what it has scanned as an upload part and resumes scanning the log from where it left off.

This option has a short form and long form: you can use **inc** or **Increment**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

Example

The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "inc=32000"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION inc='32k';
```

LockTables (lt) extended option

Function	Specifies that articles (table or parts of tables in the publications being synchronized) should be locked before synchronizing.
Syntax	dbmlsync -e lt= { ON OFF SHARE EXCLUSIVE }; ...
Description	SHARE means that dbmlsync locks all synchronization tables in shared mode. EXCLUSIVE means that dbmlsync locks all synchronization tables in exclusive mode. For all platforms except Windows CE, ON is the same as SHARE. For Windows CE devices, ON is the same as EXCLUSIVE. The default is ON .

Set to OFF to allow modifications during synchronization.

☞ For more information about shared and exclusive locks, see “How locking works” [ASA SQL User’s Guide, page 131] and “LOCK TABLE statement” [ASA SQL Reference, page 493].

☞ For more information about locking tables in MobiLink applications, see “Concurrency during synchronization” [MobiLink Synchronization User’s Guide, page 187].

When synchronization tables are locked in exclusive mode (the default for Windows CE devices), no other connections can access the tables, and so dbmlsync stored procedures that require a separate connection will not be able to execute if they require access to any of the synchronization tables. The stored procedures that require a separate connection are

- ◆ sp_hook_dbmlsync_download_com_error
- ◆ sp_hook_dbmlsync_download_fatal_sql_error
- ◆ sp_hook_dbmlsync_download_log_ri_violation

This option has a short form and long form: you can use **lt** or **LockTables**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

Example

The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "lt=on"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION  
  TO sales_publication  
  FOR ml_user1  
  OPTION lt='on' ;
```

Memory (mem) extended option

Function

Specifies a cache size.

Syntax

dbmlsync -e mem= *number* [**K** | **M**]; ...

Description

Specifies the memory used for building the upload stream, in units of bytes. Use the suffix k or m to specify units of kilobytes or megabytes, respectively. The default is **1M**.

This option has a short form and long form: you can use **mem** or **Memory**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

Example

The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "mem=2M"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION  
  TO sales_publication  
  FOR ml_user1  
  OPTION mem='2m' ;
```

MobiLinkPwd (mp) extended option

Function

Specifies the MobiLink password.

Syntax

dbmlsync -e mp=*password*; ...

Description

Specifies the password used to connect. This password should be the correct password for the MobiLink user whose subscriptions are being synchronized. This user may be specified with the dbmlsync -u option. The

default is **NULL**.

If the MobiLink user already has a password, use the extended option **-e mn** to change it.

This option has a short form and long form: you can use **mp** or **MobiLinkPwd**.

You can also store extended options in the database. For more information about dbmlsync extended options, see “[-e extended options](#)” on page 44.

See also “[NewMobiLinkPwd \(mn\) extended option](#)” on page 57
 “[-mn option](#)” on page 73
 “[-mp option](#)” on page 73

Example The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "mp=SQL"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION mp='SQL' ;
```

NewMobiLinkPwd (mn) extended option

Function	Specifies a new password.
Syntax	dbmlsync -e mn=new-password; ...
Description	<p>Specifies a password for the MobiLink user whose subscriptions are being synchronized. Use this option when you want to change an existing password. The default is NULL.</p> <p>This option has a short form and long form: you can use mn or NewMobiLinkPwd.</p> <p>You can also store extended options in the database. For more information about dbmlsync extended options, see “-e extended options” on page 44.</p>
See also	<p>“MobiLinkPwd (mp) extended option” on page 56</p> <p>“-mn option” on page 73</p> <p>“-mp option” on page 73</p>

Example

The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "mp=oldpassword; mn=newpassword"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION  
  TO sales_publication  
  FOR ml_user1  
  OPTION mn='SQL' ;
```

OfflineDirectory (dir) extended option

Function

Specifies the path containing offline transaction logs.

Syntax

dbmlsync -e dir=path; ...

Description

This option has a short form and long form: you can use **dir** or **OfflineDirectory**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

Example

The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "dir=c:\db\logs"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION  
  TO sales_publication  
  FOR ml_user1  
  OPTION dir='c:\db\logs' ;
```

PollingPeriod (pp) extended option

Function

Specifies the logscan polling period.

Syntax

dbmlsync -e pp=number [S | M | H | D]; ...

Description

This option specifies the interval between log scans. Use the suffix s, m, h, or d to specify seconds, minutes, hours or days, respectively. The default is 1 minute. If you do not specify a suffix, the default unit of time is minutes.

Logscan polling occurs only when you are scheduling synchronizations or using the sp_hook_dbmlsync_delay hook.

☞ For an explanation of logscan polling, see [“DisablePolling \(p\) extended option” on page 47](#).

This option is identical to **dbmlsync -pp**.

This option has a short form and long form: you can use **pp** or **PollingPeriod**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

See also

[“DisablePolling \(p\) extended option” on page 47](#)

[“-pp option” on page 77](#)

[“-p option” on page 75](#)

[“sp_hook_dbmlsync_delay” on page 273](#)

Example

The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "pp=5"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION pp='5';
```

Schedule (sch) extended option

Function

Specifies a schedule for synchronization.

Syntax

dbmlsync -e sch=schedule; ...

schedule = { **EVERY**:*hhhh:mm* | *singleSchedule* | **INFINITE**,... }

hhhh : **00**... **9999**

mm : **00**... **59**

singleSchedule : *day* @*hh:mm* [**AM** | **PM**] [-*hh:mm* [**AM** | **PM**]]

hh : **00**... **24**

mm : **00**... **59**

day :

EVERYDAY | **WEEKDAY** | **MON** | **TUE** | **WED** | **THU** | **FRI** | **SAT** | **SUN** | *day-OfMonth*

dayOfMonth : 0 . . . 31

Parameters

EVERY The EVERY keyword causes synchronization to occur immediately, and then repeat indefinitely after the specified time period. If the synchronization process takes longer than the specified period, synchronization starts again immediately.

singleSchedule Given one or more single schedules, synchronization occurs only at the specified days and times.

An interval is specified as @*hh:mm-hh:mm* (with optional specification of AM or PM). If AM or PM is not specified, a 24-hour clock is assumed. 24:00 is interpreted as 00:00 on the next day. When an interval is specified, synchronization occurs, starting at a random time within the interval. The interval provides a window of time for synchronization so that multiple remote databases with the same schedule do not cause congestion at the synchronization server by synchronizing at exactly the same time.

The interval end time is always interpreted as following the start time. When the time interval includes midnight, it ends on the next day. If dbmlsync is started midway through the interval, synchronization occurs at a random time before the end time.

EVERYDAY EVERYDAY is all seven days of the week.

WEEKDAY WEEKDAY is Monday through Friday.

Days of the week are Mon, Tue, and so on. You may also use the full forms of the day, such as Monday. You must use the full forms of the day names if the language you are using is not English, is not the language requested by the client in the connection string, and is not the language which appears in the server window.

dayOfMonth To specify the last day of the month regardless of the length of the month, set the *dayOfMonth* to 0.

Description

If a previous synchronization is still incomplete at a scheduled time, the scheduled synchronization commences when the previous synchronization completes.

The default is no schedule.

This option has a short form and long form: you can use **sch** or **Schedule**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

The schedule option syntax is the same when used in the synchronization SQL statements and in the dbmlsync command line.

The IgnoreScheduling extended option and the -is option instruct dbmlsync to ignore scheduling, so that synchronization is immediate. For more information, see [“-is option” on page 72](#).

☞ For more information about scheduling, see “Scheduling synchronization” [*MobiLink Synchronization User’s Guide*, page 198].

Example

The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "sch=WEEKDAY@8:00am,SUN@9:00pm"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION sch='WEEKDAY@8:00am,SUN@9:00pm' ;
```

ScriptVersion (sv) extended option

Function Specifies a script version.

Syntax **dbmlsync -e sv=version-name; ...**

Description The script version determines which scripts are run by MobiLink on the consolidated database during synchronization. The default script version name is **default**.

This option has a short form and long form: you can use **sv** or **ScriptVersion**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

Example

The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "sv=SyaAd001"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION sv='SysAd001' ;
```

SendColumnNames (scn) extended option

Function	Specifies that column names should be sent in the upload.
Syntax	dbmlsync -e scn= { ON OFF }; ...
Description	<p>Set this option to ON to tell dbmlsync to send column names from the remote database to the server. This option is required when you generate scripts automatically using the dbmlsrv9 -za or -ze options. This option increases the size of your upload, so you probably won't want to use it if you are not using -za or -ze.</p> <p>The default is OFF.</p> <p>This option has a short form and long form: you can use scn or SendColumnNames.</p> <p>You can also store extended options in the database. For more information about dbmlsync extended options, see “-e extended options” on page 44.</p>
See also	<p>“-za option” on page 28</p> <p>-ze option</p>
Example	<p>The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:</p> <pre>dbmlsync -e "scn=on"</pre> <p>The following SQL statement illustrates how you can store this option in the database:</p> <pre>CREATE SYNCHRONIZATION SUBSCRIPTION TO sales_publication FOR ml_user1 OPTION scn='on' ;</pre>

SendDownloadACK (sa) extended option

Function	Specifies that a download acknowledgement should be sent from the client to the server.
Syntax	dbmlsync -e sa= { ON OFF }; ...
Description	Turning the acknowledgement off (the default) can lead to less contention in the consolidated database and also increased throughput due to shorter download transactions. Download transactions are shorter because they are committed or rolled back as soon as possible, since MobiLink doesn't need to keep these transactions open for as long as it takes the remote client to

apply the download. Enable client side download buffering to get the most performance out of eliminating the download acknowledgement. It is recommended that `SendDownloadAck` be set to **OFF**.

☞ For more information about improving performance by turning off the download acknowledgement, see “Performance tips” [*MobiLink Synchronization User’s Guide*, page 286].

Note: When `SendDownloadAck` is set to **ON** and you are in verbose mode, an acknowledgement line is written to the client log.

The default is **OFF**.

This option has a short form and long form: you can use **sa** or **SendDownloadACK**.

You can also store extended options in the database. For more information about `dbmlsync` extended options, see “[-e extended options](#)” on page 44.

Example

The following `dbmlsync` command line illustrates how you can set this option when you start `dbmlsync`:

```
dbmlsync -e "sa=off"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION sa='off';
```

SendTriggers (st) extended option

Function Specifies that trigger actions should be sent on upload.

Syntax **dbmlsync -e st= { ON | OFF }; ...**

Description Cascaded deletes are also considered trigger actions.

The default is **OFF**.

This option has a short form and long form: you can use **st** or **SendTriggers**.

You can also store extended options in the database. For more information about `dbmlsync` extended options, see “[-e extended options](#)” on page 44.

Example

The following `dbmlsync` command line illustrates how you can set this option when you start `dbmlsync`:

```
dbmlsync -e "st=on"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION st='on' ;
```

TableOrder (tor) extended option

Function Specifies the order of tables in the upload stream.

Syntax **dbmlsync -e tor= tables ; ...**

Description This option allows you to specify the order of tables that are to be uploaded. Specify tables as a comma-separated list. You must specify all tables that are to be uploaded. If you list tables that are not included in the synchronization, they are ignored.

Specify table order to ensure referential integrity. For example, if Table1 refers to Table2, then Table2 must be uploaded before Table1.

In the specified table order, no table may have a foreign key that refers to a table that comes after it in the table order, unless your tables have a cyclical foreign key relationship. By default, dbmlsync selects a table order that satisfies this condition.

There are no cases where this option must be used. It is provided for users who for some reason (usually because it makes their scripts simpler on the consolidated database) would like to ensure that tables are uploaded in a specific order.

This option has a short form and long form: you can use **tor** or **TableOrder**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

See also “Referential integrity and synchronization” [*MobiLink Synchronization User’s Guide*, page 28]

Example The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "tor=admin,parent,child"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION tor='admin,parent,child';
```

UploadOnly (uo) extended option

Function	Specifies that synchronization should only include an upload.
Syntax	dbmlsync -e uo={ ON OFF } ; ...
Description	<p>During an upload only synchronization, dbmlsync prepares and sends an upload to the MobiLink synchronization server exactly as in a normal full synchronization. However, instead of sending a download stream back down, MobiLink sends only an acknowledgment indicating if the upload was successfully committed.</p> <p>The default is OFF.</p> <p>This option has a short form and long form: you can use uo or UploadOnly.</p> <p>You can also store extended options in the database. For more information about dbmlsync extended options, see “-e extended options” on page 44.</p>
Example	<p>The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:</p>

```
dbmlsync -e "uo=on"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION uo='on';
```

Verbose (v) extended option

Function	Specifies full verbosity.
Syntax	dbmlsync -e v= { ON OFF } ; ...
Description	<p>This option specifies a high level of verbosity, which may affect performance and should normally be used in the development phase only.</p> <p>This option is identical to dbmlsync -v+. If you specify both -v and the extended options and there are conflicts, the -v option overrides the extended option. If there is no conflict, the verbosity logging options are additive—all</p>

options that you specify are used. When logging verbosity is set by extended option, the logging does not take effect immediately, so startup information is not logged. By the time of the first synchronization, the logging behavior is identical between the `-v` options and the extended options.

☞ For more information, see [“-v option” on page 80](#).

The default is **OFF**.

This option has a short form and long form: you can use **v** or **Verbose**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

Example

The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "v=on"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION  
  TO sales_publication  
  FOR ml_user1  
  OPTION v='on';
```

VerboseHooks (vs) extended option

Function	Specifies that messages related to hook scripts should be logged.
Syntax	dbmlsync -e vs= { ON OFF } ; ...
Description	This option is identical to dbmlsync -vs . If you specify both <code>-v</code> and the extended options and there are conflicts, the <code>-v</code> option overrides the extended option. If there is no conflict, the verbosity logging options are additive—all options that you specify are used. When logging verbosity is set by extended option, the logging does not take effect immediately, so startup information is not logged. By the time of the first synchronization, the logging behavior is identical between the <code>-v</code> options and the extended options.

☞ For more information, see [“-v option” on page 80](#).

The default is **OFF**.

This option has a short form and long form: you can use **vs** or **VerboseHooks**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

See also [“Client event-hook procedures” on page 269](#)

Example The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "vs=on"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION vs='on' ;
```

VerboseMin (vm) extended option

Function Specifies that a small amount of information should be logged.

Syntax **dbmlsync -e vm= { ON | OFF } ; ...**

Description This option is identical to **dbmlsync -v**. If you specify both -v and the extended options and there are conflicts, the -v option overrides the extended option. If there is no conflict, the verbosity logging options are additive—all options that you specify are used. When logging verbosity is set by extended option, the logging does not take effect immediately, so startup information is not logged. By the time of the first synchronization, the logging behavior is identical between the -v options and the extended options.

☞ For more information, see [“-v option” on page 80](#).

The default is **OFF**.

This option has a short form and long form: you can use **vm** or **VerboseMin**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

Example The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "vm=on"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION vm='on' ;
```

VerboseOptions (vo) extended option

Function	Specifies that information should be logged about the command line options (including extended options) that you have specified.
Syntax	dbmlsync -e vo= { ON OFF } ; ...
Description	This option is identical to dbmlsync -vo . If you specify both -v and the extended options and there are conflicts, the -v option overrides the extended option. If there is no conflict, the verbosity logging options are additive—all options that you specify are used. When logging verbosity is set by extended option, the logging does not take effect immediately, so startup information is not logged. By the time of the first synchronization, the logging behavior is identical between the -v options and the extended options.

☞ For more information, see [“-v option” on page 80](#).

The default is **OFF**.

This option has a short form and long form: you can use **vo** or **VerboseOptions**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

Example	The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:
---------	--

```
dbmlsync -e "vo=on"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION  
TO sales_publication  
FOR ml_user1  
OPTION vo='on' ;
```

VerboseRowCounts (vn) extended option

Function	Specifies that the number of rows that are uploaded and downloaded should be logged.
Syntax	dbmlsync -e vn= { ON OFF } ; ...
Description	This option is identical to dbmlsync -vn . If you specify both -v and the extended options and there are conflicts, the -v option overrides the extended option. If there is no conflict, the verbosity logging options are additive—all options that you specify are used. When logging verbosity is set by extended

option, the logging does not take effect immediately, so startup information is not logged. By the time of the first synchronization, the logging behavior is identical between the `-v` options and the extended options.

☞ For more information, see [“-v option” on page 80](#).

The default is **OFF**.

This option has a short form and long form: you can use **vn** or **VerboseRowCounts**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

Example

The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "vn=on"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION vn='on' ;
```

VerboseRowValues (vr) extended option

Function	Specifies that the values of rows that are uploaded and downloaded should be logged.
Syntax	dbmlsync -e vr= { ON OFF } ; ...
Description	This option is identical to dbmlsync -vr . If you specify both <code>-v</code> and the extended options and there are conflicts, the <code>-v</code> option overrides the extended option. If there is no conflict, the verbosity logging options are additive—all options that you specify are used. When logging verbosity is set by extended option, the logging does not take effect immediately, so startup information is not logged. By the time of the first synchronization, the logging behavior is identical between the <code>-v</code> options and the extended options.

☞ For more information, see [“-v option” on page 80](#).

The default is **OFF**.

This option has a short form and long form: you can use **vr** or **VerboseRowValues**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

Example

The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "vr=on"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION  
  TO sales_publication  
  FOR ml_user1  
  OPTION vr='on' ;
```

VerboseUpload (vu) extended option

Function

Specifies that information about the upload steam should be logged.

Syntax

dbmlsync -e vu= { ON | OFF } ; ...

Description

This option is identical to **dbmlsync -vu**. If you specify both -v and the extended options and there are conflicts, the -v option overrides the extended option. If there is no conflict, the verbosity logging options are additive—all options that you specify are used. When logging verbosity is set by extended option, the logging does not take effect immediately, so startup information is not logged. By the time of the first synchronization, the logging behavior is identical between the -v options and the extended options.

☞ For more information, see [“-v option” on page 80](#).

The default is **OFF**.

This option has a short form and long form: you can use **vu** or **VerboseUpload**.

You can also store extended options in the database. For more information about dbmlsync extended options, see [“-e extended options” on page 44](#).

Example

The following dbmlsync command line illustrates how you can set this option when you start dbmlsync:

```
dbmlsync -e "vu=on"
```

The following SQL statement illustrates how you can store this option in the database:

```
CREATE SYNCHRONIZATION SUBSCRIPTION  
  TO sales_publication  
  FOR ml_user1  
  OPTION vu='on' ;
```

-eh option

Function	Ignores errors that occur in hook functions.
Syntax	dbmlsync -eh ...

-ek option

Function	Allows you to specify the encryption key for strongly encrypted databases directly on the command line.
Syntax	dbmlsync -ek <i>key</i> ...
Description	If you have a strongly encrypted database, you must provide the encryption key to use the database or transaction log in any way, including offline transactions. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify a key for a strongly encrypted database.

-ep option

Function	Prompt for the encryption key.
Syntax	dbmlsync -ep ...
Description	This option causes a dialog box to appear, in which you enter the encryption key. It provides an extra measure of security by never allowing the encryption key to be seen in clear text. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify a key for a strongly encrypted database.

-eu option

Function	Specifies extended upload options.
Syntax	dbmlsync -n <i>publication-name</i> -eu <i>keyword=value</i> ;...
Description	Extended options that are specified on the command line with the -eu option apply only to the synchronization specified by the -n option they follow. For example, on the following command line, the extended option <code>sv=test</code> applies only to the synchronization of <code>pub2</code> .

```
dbmlsync -n pub1 -n pub2 -eu sv=test
```


For an explanation of how extended options are processed when they are set in more than one place, see [“-e extended options” on page 44](#).

For a complete list of extended options, see “[-e extended options](#)” on [page 44](#).

-i option

Function	Executes the SQL script contained in the named file.
Syntax	dbmlsync -i filename ...
Description	<p>This option is intended for upgrading applications and making schema changes to deployed remote databases. Use this option when you make schema changes to ensure that all changes are in a compatible format.</p> <p>The SQL script contained in the specified file is executed immediately after synchronization is complete and before releasing the table locks. The script is executed if dbmlsync received confirmation from MobiLink that the upload was applied even if an error occurs during the download.</p> <p>You should be explicit about commit/rollback operations when using the -i option. Failure to do so may cause inconsistent results.</p>

-is option

Function	Ignores scheduling instructions so that synchronization is immediate.
Syntax	dbmlsync -is ...
Description	<p>Ignore extended options that schedule synchronization.</p> <p> For information about scheduling, see “Scheduling synchronization” [<i>MobiLink Synchronization User’s Guide</i>, page 198].</p>

-k option

Function	Closes window on completion.
Syntax	dbmlsync -k ...
Description	Close window on completion, if used together with the -o option.

-l option

Function	Lists available extended options.
Syntax	dbmlsync -l ...
Description	When used with the dbmlsync command line it shows you available extended options.

-mn option

Function	Supplies a new password for the user being synchronized.
Syntax	dbmlsync -mn password ...
Description	<p>Changes the MobiLink user's password.</p> <p>☞ For more information, see “Authenticating MobiLink Users” [<i>MobiLink Synchronization User's Guide</i>, page 103].</p>
See also	<p>“MobiLinkPwd (mp) extended option” on page 56</p> <p>“NewMobiLinkPwd (mn) extended option” on page 57</p> <p>“-mp option” on page 73</p>

-mp option

Function	Supplies the password of the user being synchronized.
Syntax	dbmlsync -mp password ...
Description	<p>Supplies the password for MobiLink user authentication.</p> <p>☞ For more information, see “Authenticating MobiLink Users” [<i>MobiLink Synchronization User's Guide</i>, page 103].</p>
See also	<p>“MobiLinkPwd (mp) extended option” on page 56</p> <p>“NewMobiLinkPwd (mn) extended option” on page 57</p> <p>“-mn option” on page 73</p>

-n option

Function	Names the synchronization publication.
Syntax	dbmlsync -n pubname ...
Description	<p>Name of synchronization publication. You can supply more than one -n option to synchronize more than one synchronization publication.</p> <p>There are two ways to use -n to synchronize multiple publications:</p> <ul style="list-style-type: none">◆ Specify <code>-n pub1 , pub2 , pub3</code> to upload pub1, pub2, and pub3 in one upload stream. <p>In this case, if you have set extended options on the publications, only the options set on the first publication in the list are used. Extended options set on subsequent publications are ignored.</p>

-
- ◆ Specify `-n pub1 -n pub2 -n pub3` to upload `pub1` in one upload stream, `pub2` in another, and `pub3` in a third upload stream.

When successive synchronizations occur very quickly, such as when you specify `-n pub1 -n pub2`, it is possible that `dbmlsync` may start processing a synchronization when the server is still processing the previous synchronization. In this case, the second synchronization will fail with an error indicating that concurrent synchronizations are not allowed. If you run into this situation, you can define an `sp_hook_dbmlsync_delay` stored procedure to create a delay before each synchronization. Usually a few seconds to a minute is a sufficient delay.

☞ For more information, see [“sp_hook_dbmlsync_delay” on page 273](#).

-o option

Function	Sends output to a log file.
Syntax	dbmlsync -o <i>filename</i> ...
Description	Append output to a log file. Default is to send output to the screen.
See also	“-os option” on page 74 “-ot option” on page 75

-os option

Function	Specifies the maximum size of the output log.
Syntax	dbmlsync -os <i>size</i> [K M G]. ...
Description	<p>The <i>size</i> is the maximum file size for logging output messages, specified in units of bytes. Use the suffix <code>k</code>, <code>m</code> or <code>g</code> to specify units of kilobytes, megabytes or gigabytes, respectively. By default, there is no size limit. The minimum size limit is 10 kb.</p> <p>Before the <code>dbmlsync</code> utility logs output messages to a file, it checks the current file size. If the log message will make the file size exceed the specified size, the <code>dbmlsync</code> utility renames the output file to <code>yymmddxx.dbr</code>, where <code>yymmdd</code> represents the year, month, and day, and <code>xx</code> are sequential characters ranging from <code>AA</code> to <code>ZZ</code>.</p> <p>This option allows you to manually delete old log files and free up disk space.</p>
See also	“-o option” on page 74 “-ot option” on page 75

-ot option

Function	Truncates the log file and appends output messages to it.
Syntax	dbmlsync -ot logfile ...
Description	The functionality is the same as the -o option except the log file is truncated before any messages are written to it.
See also	“-o option” on page 74 “-os option” on page 74

-p option

Function	Disables logscan polling.
Syntax	dbmlsync -p ...
Description	<p>In order to build an upload stream, dbmlsync must scan the transaction log. Usually it does this just before synchronization. However, when synchronizations are scheduled or when the <code>sp_hook_dbmlsync_delay</code> hook is used, dbmlsync by default scans the log in the pause that occurs just before synchronization. This behavior is more efficient because when synchronization begins the log is already at least partially scanned. This default behavior is called logscan polling.</p> <p>Logscan polling is on by default but only has an effect when synchronizations are scheduled using scheduling options or when <code>sp_hook_dbmlsync_delay</code> hook is used. When in effect, polling occurs at set intervals; by default this is 1 minute, but it can be changed with the <code>dbmlsync -pp</code> option.</p> <p>The default is to not disable logscan polling (OFF).</p> <p>This option is identical to dbmlsync -e p.</p>
See also	“DisablePolling (p) extended option” on page 77 “PollingPeriod (pp) extended option” on page 58 “-pp option” on page 77

-pd option

Function	Preload specified dlls for Windows CE.
Syntax	dbmlsync -pd dllname;...

Description When running dbmlsync on Windows CE, you should use the -pd option to specify dlls that need to be loaded. Otherwise, the correct dlls may not be loaded and an error may be generated.

Following are the dlls that need to be loaded for each communication protocol:

Protocol	DLL
TCP/IP	dbmlsock9.dll
HTTP	dbmlhttp9.dll
HTTPS	dbmlhttps9.dll

You should specify multiple dlls as a semicolon-separated list. For example,

```
-pd dbmlsock9.dll;dbmlhttp9.dll
```

-pi option

Function Pings a MobiLink synchronization server.

Syntax **dbmlsync -pi -c *connection_string* -e *sv=script_version***
[-n *pubname*] [-u *ml_username*]

Description The ping option allows you to test that your connection information is correct. When you use -pi, dbmlsync does not initiate synchronization.

In order to be able to ping, dbmlsync must have a unique address for the MobiLink synchronization server. This means that you must include connection parameters and a script version, as well as the publication name, MobiLink user name, or both. The publication or user hold connection information for the remote. You need to specify both when the user is subscribed to multiple publications or the publication has multiple users. For example, if there is only one subscription to the publication, you can specify the publication without the user.

When the MobiLink synchronization server receives a ping request, it connects to the consolidated database, authenticates the user, and then sends the authenticating user status and value back to the client (dbmlsync or UltraLite).

If the ping succeeds, the MobiLink server issues an information message. If the ping does not succeed, it issues an error message.

If the MobiLink user name cannot be found in the ml_user system table and the MobiLink server is running with the command line option -zu+, the MobiLink server adds the user to ml_user.

The MobiLink synchronization server may execute the following scripts, if they exist:

- ◆ begin_connection
- ◆ authenticate_user
- ◆ authenticate_user_hashed
- ◆ end_connection

The client cannot synchronize while it is pinging the server.

-pp option

Function	Specifies the frequency of log scans.
Syntax	dbmlsync -pp <i>number</i> [h m s] . . .
Description	<p>This option specifies the interval between log scans. Use the suffix s, m, h, or d to specify seconds, minutes, hours or days, respectively. The default is 1 minute. If you do not specify a suffix, the default unit of time is minutes.</p> <p>Logscan polling occurs only when you are scheduling synchronizations or using the sp_hook_dbmlsync_delay hook.</p> <p>☞ For an explanation of logscan polling, see “-p option” on page 75.</p>
See also	<p>“PollingPeriod (pp) extended option” on page 58</p> <p>“DisablePolling (p) extended option” on page 47</p> <p>“-p option” on page 75</p>

-q option

Function	Starts the MobiLink synchronization client in a minimized window.
Syntax	dbmlsync -q . . .
Description	For Windows operating systems only.

-r option

Function	<p>Specifies that the remote offset should be used when there is disagreement between the offsets in the remote and consolidated databases.</p> <p>The -rb option can be used when the remote offset is less than the consolidated offset (such as when the remote database has been restored from backup). The -r option is provided for backward compatibility and is</p>
----------	---

identical to **-rb**. The **-ra** option, used when the remote offset is greater than the consolidated offset, is provided only for very rare circumstances and may cause data loss.

Syntax

dbmlsync { -r | -ra | -rb } ...

Description

The **offset**, also called the **progress**, refers to a position in the transaction log of the remote database. It indicates the point to which all operations for the subscription have been uploaded and acknowledged. **dbmlsync** uses the offset to decide what data to upload. On the remote database, the offset is stored in the progress column of the **SYS.SYSSYNC** system table. On the consolidated database, the offset is stored in the progress column of the **ml_user** table for version 7.x databases, and in the progress column of the **ml_subscription** table for version 8.0 and up databases.

For each remote, the remote and consolidated databases maintain an offset for every subscription. When a user synchronizes, the offsets are confirmed for all subscriptions that are associated with the user. If there is any disagreement between the remote and consolidated database offsets, the default behavior is to update the offsets on the remote with values from the consolidated and then send a new upload based on those offsets. In most cases, this default is appropriate. For example, it is generally appropriate when the consolidated database is restored from backup and the remote transaction log is intact, or when an upload is successful but communication failure prevented an upload acknowledgement from being sent.

-rb If the remote database is restored from backup, the default behavior may cause data to be lost. In this case, the first time you run **dbmlsync** after the remote database is restored, you should specify **-rb**. When you use **-rb**, the upload continues from the offset recorded in the remote database if the offset recorded in the remote is less than that obtained from the consolidated database. If you use **-rb** and the offset in the remote is not less than the offset from the consolidated database, an error is reported and the synchronization is aborted.

The **-rb** option may result in some data being uploaded that has already been uploaded. This can result in conflicts in the consolidated database and should be handled with appropriate conflict resolution scripts.

-ra The **-ra** option should be used only in very rare cases. If you use **-ra**, the upload is retried starting from the offset obtained from the remote database if the remote offset is greater than the offset obtained from the consolidated database. If you use **-ra** and the offset in the remote is not greater than the offset from the consolidated database, an error is reported and the synchronization is aborted.

The **-ra** option should be used with care. If the offset mismatch is the result

of a restore of the consolidated database, changes that happened in the remote database in the gap between the two offsets are lost. The `-ra` option may be useful when the consolidated database has been restored from backup and the remote database transaction log has been truncated at the same point as the remote offset. In this case, all data that was uploaded from the remote database is lost from the point of the consolidated offset to the point of the remote offset.

-sc option

Function	Specifies that <code>dbmlsync</code> should reload schema information before each synchronization.
Syntax	dbmlsync -sc...
Description	<p>Prior to version 9.0, <code>dbmlsync</code> reloaded schema information from the database before each synchronization. The information that was reloaded includes foreign key relationships, publication definitions, extended options stored in the database, and information about database settings. On slower handheld devices, loading this information typically took 20 seconds. In most cases this information does not change between synchronizations.</p> <p>Starting with version 9.0, by default <code>dbmlsync</code> loads schema information only at startup. Specify <code>-sc</code> if you want the information to be loaded before every synchronization.</p>

-u option

Function	Specifies the MobiLink user name.
Syntax	dbmlsync -u <i>ml_username</i> ...
Description	<p>You can specify one user in the <code>dbmlsync</code> command line, where <i>ml_username</i> is the name used in the FOR clause of the CREATE SYNCHRONIZATION SUBSCRIPTION statement corresponding to the subscription to be processed.</p> <p>This option should be used in conjunction with <code>-n <i>publication</i></code> to identify the subscription on which <code>dbmlsync</code> should operate. Each subscription is uniquely identified by an <i>ml_username</i>, <i>publication</i> pair.</p> <p>You can only specify one user name on the command line. All subscriptions to be synchronized in a single run must involve the same user. The <code>-u</code> option can be omitted if each publication that is specified on the command line with the <code>-n</code> option has only one subscription.</p>

-uo option

Function	Specifies that synchronization will only include an upload, and no download will occur.
Syntax	dbmlsync -uo . . .
Description	During an upload only synchronization, dbmlsync prepares and sends an upload to MobiLink exactly as it would in a normal full synchronization. However, instead of sending a download stream back down, MobiLink sends only an acknowledgment indicating if the upload was successfully committed.

-urc option

Function	Specifies an estimate of the number of rows to be uploaded in a synchronization.
Syntax	dbmlsync -urc <i>row-estimate</i> . . .
Description	<p>To improve performance, you can specify an estimate of the number of rows that will be uploaded in a synchronization. In general, a higher estimate results in faster uploads but more memory usage.</p> <p>Synchronization will proceed correctly regardless of the estimate that is specified.</p>

-v option

Function	Allows you to specify what information is logged to the message log file and displayed in the synchronization window. A high level of verbosity may affect performance and should normally be used in the development phase only.
Syntax	dbmlsync -v [<i>levels</i>] . . .
Description	<p>The -v options affect the message log file and synchronization window. You only have a message log if you specify -o or -ot on the dbmlsync command line.</p> <p>If you specify -v alone, a small amount of information is logged.</p> <p>The values of <i>levels</i> are as follows. You can use one or more of these options at once; for example, -vnrsu or -v+cp.</p> <ul style="list-style-type: none">◆ + Turn on all logging options except for c and p.◆ c Expose the connect string in the log.

- ◆ **p** Expose the password in the log.
- ◆ **n** Log the number of rows that were uploaded and downloaded.
- ◆ **o** Log information about the command line options and extended options that you have specified.
- ◆ **r** Log the values of rows that were uploaded and downloaded.
- ◆ **s** Log messages related to hook scripts.
- ◆ **u** Log information about the upload stream.

There are extended options that have similar functionality to the -v options. If you specify both -v and the extended options and there are conflicts, the -v option overrides the extended option. If there is no conflict, the verbosity logging options are additive—all options that you specify are used. When logging verbosity is set by extended option, the logging does not take effect immediately, so startup information is not logged. By the time of the first synchronization, the logging behavior is identical between the -v options and the extended options.

See also

[“Verbose \(v\) extended option” on page 65](#)

[“VerboseHooks \(vs\) extended option” on page 66](#)

[“VerboseMin \(vm\) extended option” on page 67](#)

[“VerboseOptions \(vo\) extended option” on page 68](#)

[“VerboseRowCounts \(vn\) extended option” on page 68](#)

[“VerboseRowValues \(vr\) extended option” on page 69](#)

-wc option

Function

For Windows CE only, this option specifies a Windows class name for use with ActiveSync synchronization.

Syntax

dbmlsync -wc *class-name* ...

Description

This option specifies a class name that identifies the application for ActiveSync synchronization. The class name must be given when registering the application for use with ActiveSync synchronization.

See also

[“Registering Adaptive Server Anywhere clients for ActiveSync” \[MobiLink Synchronization User’s Guide, page 192\]](#)

[“Using ActiveSync synchronization” \[MobiLink Synchronization User’s Guide, page 189\]](#)

-x option

Function	Renames and restarts the transaction log after it has been scanned for outgoing messages.
Syntax	dbmsync -x [size [K M G]. . .
Description	<p>The optional <i>size</i> means that the transaction log is renamed only if it is larger than the specified size. Use the suffix k, m or g to specify units of kilobytes, megabytes or gigabytes, respectively. The default size is 0.</p> <p>In some circumstances, synchronizing data to a consolidated database can take the place of backing up remote databases, or renaming the transaction log when the database server is shut down.</p> <p>If backups are not routinely performed at the remote database, the transaction log continues to grow. As an alternative to using the -x option to control transaction log size, you can use an Adaptive Server Anywhere event handler to control the size of the transaction log. For example, the following event handler renames the transaction log at the remote database when its size exceeds 5 Mb. You can use such an event handler together with the DELETE_OLD_LOGS database option to control the space taken up by transaction logs.</p> <pre>CREATE EVENT RenameLogLimit TYPE GrowLog WHERE event_condition('LogSize') > 5 AT REMOTE HANDLER BEGIN BACKUP DATABASE DIRECTORY backupdir TRANSACTION LOG ONLY TRANSACTION LOG RENAME END</pre> <p>See also</p> <p>“Automating Tasks Using Schedules and Events” [ASA Database Administration Guide, page 267]</p> <p>“DELETE_OLD_LOGS option [replication]” [ASA Database Administration Guide, page 590]</p>

CHAPTER 3

Synchronization Events

About this chapter

This chapter provides information about the MobiLink synchronization events and the SQL scripts, Java methods, or .NET methods that handle these events. You implement scripts to handle one or more of these events to control the actions of the MobiLink synchronization server.

☞ For information about storing scripts, see “Adding and deleting scripts in your consolidated database” [*MobiLink Synchronization User’s Guide*, page 51].

Contents

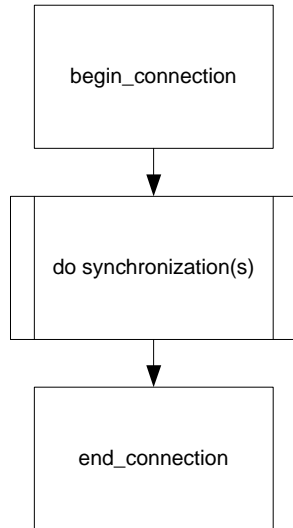
Topic:	page
Overview of MobiLink events	86
authenticate_parameters connection event	98
authenticate_user connection event	100
authenticate_user_hashed connection event	104
begin_connection connection event	107
begin_connection_autocommit connection event	109
begin_download connection event	110
begin_download table event	112
begin_download_deletes table event	114
begin_download_rows table event	116
begin_publication connection event	118
begin_synchronization connection event	121
begin_synchronization table event	123
begin_upload connection event	125
begin_upload table event	127
begin_upload_deletes table event	129
begin_upload_rows table event	131
download_cursor cursor event	133

Topic:	page
download_delete_cursor cursor event	136
download_statistics connection event	139
download_statistics table event	142
end_connection connection event	145
end_download connection event	147
end_download table event	149
end_download_deletes table event	151
end_download_rows table event	153
end_publication connection event	155
end_synchronization connection event	158
end_synchronization table event	160
end_upload connection event	162
end_upload table event	164
end_upload_deletes table event	166
end_upload_rows table event	168
example_upload_cursor table event	170
example_upload_delete table event	171
example_upload_insert table event	172
example_upload_update table event	173
handle_error connection event	174
handle_odbc_error connection event	177
modify_last_download_timestamp connection event	180
modify_next_last_download_timestamp connection event	182
modify_user connection event	184
new_row_cursor cursor event (deprecated)	186
old_row_cursor cursor event (deprecated)	189
prepare_for_download connection event	192
report_error connection event	194

Topic:	page
report_odbc_error connection event	196
resolve_conflict table event	199
synchronization_statistics connection event	202
synchronization_statistics table event	205
time_statistics connection event	207
time_statistics table event	209
upload_cursor cursor event (deprecated)	212
upload_delete table event	214
upload_fetch table event	216
upload_insert table event	218
upload_new_row_insert table event	220
upload_old_row_insert table event	222
upload_statistics connection event	224
upload_statistics table event	227
upload_update table event	231

Overview of MobiLink events

When a synchronization request occurs and MobiLink server decides that a new connection must be created, the `begin_connection` event is fired and synchronization starts.



Following the synchronization, the connection is placed in a connection pool, and MobiLink again waits for a synchronization request for the current script version. Before a connection is eventually dropped from the connection pool, the `end_connection` event is fired. But if another synchronization request for the same version is received, then MobiLink handles the next synchronization request on the same connection. There are a number of events that affect the current synchronization.

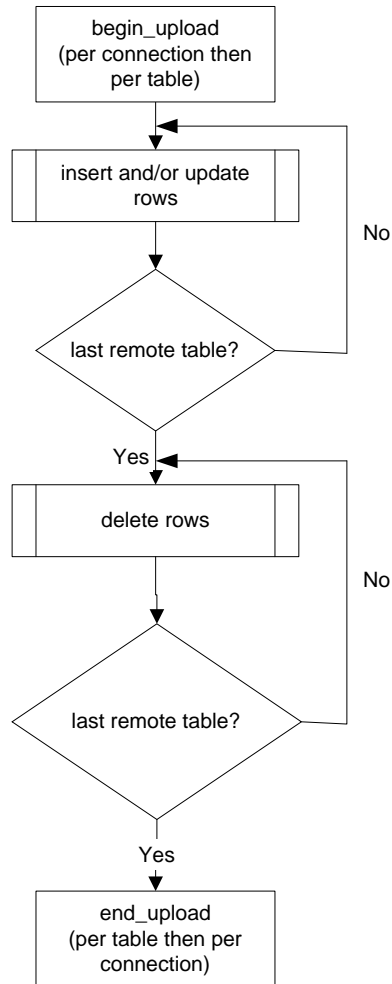
The primary phases of a synchronization are the upload and download transactions. The events contained in the upload and download transactions are outlined below.

The upload transaction

The upload transaction applies changes uploaded from a remote database.

The `begin_upload` event marks the beginning of the upload transaction. The upload transaction is a two-part process. First, inserts and updates are uploaded for all remote tables, and second, deletes are uploaded for all remote tables.

upload transaction



The end_upload event marks the end of the upload transaction.

For more information about the events that happen during upload, see “Writing scripts to upload rows” [*MobiLink Synchronization User’s Guide*, page 54].

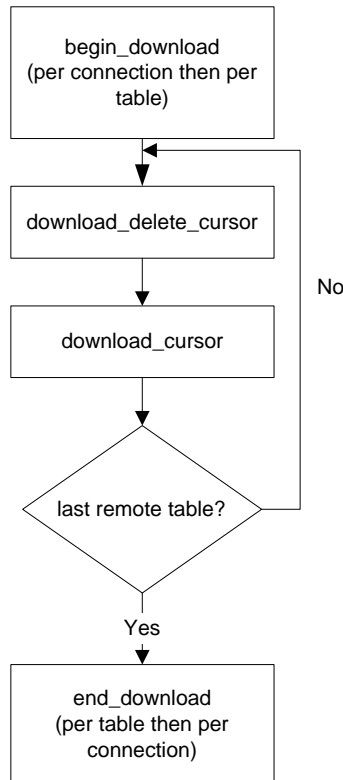
The download
transaction

The download transaction fetches rows from the consolidated database. It

begins with the `begin_download` event.

The download transaction is a two-part process. For each table, first deletes are downloaded, and then update/insert rows (upserts) are downloaded. The `end_download` event ends the download transaction.

download transaction



☞ For more information about the events that happen during download, see “Writing scripts to download rows” [*MobiLink Synchronization User’s Guide*, page 56].

The following pseudo code provides an overview of the sequence in which events, and hence the script of the same name, are invoked.

Event overview in
pseudo-code

The following pseudo-code shows the complete MobiLink synchronization event model. This model assumes a full synchronization (not upload-only or download-only) with no errors.

Notes

- ◆ In most cases, if you have not defined a script for a given event, the default action is to do nothing.
- ◆ The `begin_connection` and `end_connection` events are **connection-level events**. They are independent of any single synchronization and have no parameters.
- ◆ Some events are invoked once per synchronization for each table being synchronized. Scripts associated with these events are called **table-level scripts**.

While each table can have its own table scripts, you can also write table-level scripts that are shared by several tables.

- ◆ Some events, such as `begin_synchronization`, occur at both the connection level and the table level. You can supply both connection and table scripts for these events.
- ◆ The COMMIT statements illustrate how the synchronization process is broken up into distinct transactions.
- ◆ A database error can occur at any point within the synchronization process. Database errors are handled using the `handle_error` or `handle_odbc_error` scripts.

Warning

There should be no implicit or explicit commit or rollback in your synchronization scripts or the procedures or triggers that are called from your synchronization scripts. COMMIT or ROLLBACK statements within scripts alter the transactional nature of the synchronization steps. If you use them, you cannot guarantee the integrity of your data in the event of a failure.

Synchronization events in pseudo-code.

Legend:

- `// This is a comment`
- `<name>`
The pseudo code for `<name>` is listed separately in a later section, under a banner:

name

- `VariableName <- value`
Assign the given value to the given variable name. Variable names are in mixed case.
- `event_name`
If you have defined a script for the given event name, it will be invoked.

```
CONNECT to consolidated database
begin_connection_autocommit
begin_connection
COMMIT
for each synchronization request with
    the same script version {
    <synchronize>
}
end_connection
COMMIT
DISCONNECT from consolidated database
```

```
-----
synchronize
-----
```

```
<authenticate>
<begin_synchronization>
<upload>
<prepare_for_download>
<download>
<end_synchronization>
```

```
-----
authenticate
-----
```

```
Status <- 1000
UseDefaultAuthentication <- TRUE
if( authenticate_user script is defined ) {
    UseDefaultAuthentication <- FALSE
    TempStatus <- authenticate_user
    if( TempStatus > Status ) {
        Status <- TempStatus
    }
}
```



```

if( authenticate_user_hashed script is defined ) {
    UseDefaultAuthentication <- FALSE
    TempStatus <- authenticate_user_hashed
    if( TempStatus > Status ) {
        Status <- TempStatus
    }
}
if( UseDefaultAuthentication ) {
    if( the user exists in the ml_user table ) {
        if( ml_user.hashed_password column is not NULL ) {
            if( password matches ml_user.hashed_password ) {
                Status <- 1000
            } else {
                Status <- 4000
            }
        } else {
            Status <- 1000
        }
    } else if( -zu+ was on the command line ) {
        Status <- 1000
    } else {
        Status <- 4000
    }
}
if( Status <= 2000 ) {
    if( authenticate_parameters script is defined )
    {
        TempStatus <- authenticate_parameters
        if( TempStatus > Status ) {
            Status <- TempStatus
        }
    }
}
if( Status >= 3000 ) {
    ROLLBACK
    // Abort the synchronization.
} else {
    // UserName defaults to MobiLink user name
    // sent from the remote.
    if( modify_user script is defined ) {
        UserName <- modify_user
        // The new value of UserName is later passed to
        // all scripts that expect the MobiLink user name.
    }
    COMMIT
}
}

```

```

-----
begin_synchronization
-----

begin_synchronization
for each publication being synchronized {
    begin_publication
    COMMIT
}

-----
end_synchronization
-----

for each publication being synchronized {
    if( begin_publication script was called ) {
        end_publication
    }
}
for each table being synchronized {
    if( begin_synchronization table script was called ) {
        end_synchronization // table event
    }
}
end_synchronization      // connection event
synchronization_statistics
time_statistics
COMMIT

```

☞ For the details of upload stream processing, see [“Events during upload” on page 92](#).

☞ For the details of download stream processing, see [“Events during download” on page 96](#).

Events during upload

The following pseudo-code illustrates how upload events and upload scripts are invoked.

These events take place at the upload location in the complete event model. For more information, see [“Overview of MobiLink events” on page 86](#).

```

-----
upload
-----

begin_upload
  for each table being synchronized {
    begin_upload_rows
    for each uploaded INSERT or UPDATE for this table {
      if( INSERT ) {
        <upload_inserted_row>
      }
      if( UPDATE ) {
        <upload_updated_row>
      }
    }
    end_upload_rows
  }
  for each table being synchronized IN REVERSE ORDER {
    begin_upload_deletes
    for each uploaded DELETE for this table {
      <upload_deleted_row>
    }
    end_upload_deletes
  }
end_upload
upload_statistics
COMMIT

-----
<upload_inserted_row>
-----

// NOTES:
// - Only table scripts for the current table are involved.
// - Cursor-based upload scripts, like upload_cursor,
//   are deprecated, and so are not shown.

UploadUsingStatements <- (
  upload_insert script is defined
  or upload_update script is defined
  or upload_delete script is defined
  or upload_fetch script is defined
  or upload_new_row_insert script is defined
  or upload_old_row_insert script is defined )

```

```

if( UploadUsingStatements ) {
  ConflictsAreExpected <- (
    upload_new_row_insert script is defined
    or upload_old_row_insert script is defined
    or resolve_conflict script is defined )
  if( upload_insert script is defined ) {
    upload_insert
  } else if( ConflictsAreExpected
    and upload_update script is not defined
    and upload_insert script is not defined
    and upload_delete script is not defined ) {
    // Forced conflict.
    upload_new_row_insert
    resolve_conflict
  } else {
    // ignore the insert
  }
} else {
  // ignore the insert
}

-----
upload_updated_row
-----

// NOTES:
// - Only table scripts for the current table are involved.
// - Both the old (original) and new rows are uploaded for
//   each update.
// - Cursor-based upload scripts, like upload_cursor,
//   are deprecated, and so are not shown.

```

```

UploadUsingStatements <- (
  upload_insert script is defined
  or upload_update script is defined
  or upload_delete script is defined
  or upload_fetch script is defined
  or upload_new_row_insert script is defined
  or upload_old_row_insert script is defined )
if( UploadUsingStatements ) {
  ConflictsAreExpected <- (
    upload_new_row_insert script is defined
    or upload_old_row_insert script is defined
    or resolve_conflict script is defined )
  Conflicted <- FALSE
  if( upload_update script is defined ) {
    if( ConflictsAreExpected
      and upload_fetch script is defined ) {
      FETCH using upload_fetch INTO current_row
      if( current_row <> old_row ) {
        Conflicted <- TRUE
      }
    }
    if( not Conflicted ) {
      upload_update
    }
  } else if( upload_update script is not defined
    and upload_insert script is not defined
    and upload_delete script is not defined ) {
    // Forced conflict.
    Conflicted <- TRUE
  }
  if( ConflictsAreExpected and Conflicted ) {
    upload_old_row_insert
    upload_new_row_insert
    resolve_conflict
  }
} else {
  // ignore the update
}

-----
upload_deleted_row
-----

// NOTES:
// - Only table scripts for the current table are involved.
// - Cursor-based upload scripts, like upload_cursor,
//   are deprecated, and so are not shown.

UploadUsingStatements <- (
  upload_insert script is defined
  or upload_update script is defined
  or upload_delete script is defined
  or upload_fetch script is defined
  or upload_new_row_insert script is defined
  or upload_old_row_insert script is defined )

```

```

if( UploadUsingStatements ) {
  ConflictsAreExpected <- (
    upload_new_row_insert script is defined
    or upload_old_row_insert script is defined
    or resolve_conflict script is defined )
  if( upload_delete is defined ) {
    upload_delete
  } else if( ConflictsAreExpected
    and upload_update script is not defined
    and upload_insert script is not defined
    and upload_delete script is not defined ) {
    // Forced conflict.
    upload_old_row_insert
    resolve_conflict
  } else {
    // ignore this delete
  }
} else {
  // ignore this delete
}

```

Events during download

The following pseudo-code provides an overview of the sequence in which download events, and hence the script of the same name, are invoked.

These events take place at the download location in the complete event model provided in [“Overview of MobiLink events” on page 86](#).

```

-----
prepare_for_download
-----

prepare_for_download
modify_last_download_timestamp
if( prepare_for_download script is defined
  or modify_last_download_timestamp script is defined ) {
  COMMIT
}

```

```

-----
download
-----

begin_download
for each table being synchronized {
  begin_download_deletes
  for each row in download_delete_cursor {
    if( all primary key columns are NULL ) {
      send TRUNCATE to remote
    } else {
      send DELETE to remote
    }
  }
  end_download_deletes
  begin_download_rows
  for each row in download_cursor {
    send INSERT ON EXISTING UPDATE to remote
  }
  end_download_rows
}
modify_next_download_timestamp
end_download
COMMIT

```

Notes

- ◆ If an acknowledgement is expected, and if no confirmation of the downloads is received from the client, the entire download transaction is rolled back in the consolidated database.

☞ For more information, see [“SendDownloadACK \(sa\) extended option” on page 62](#) for Adaptive Server Anywhere remotes or “Send Download Acknowledgement synchronization parameter” [*UltraLite Database User’s Guide*, page 172] for UltraLite remotes.

- ◆ The download stream does not distinguish between inserts and updates. The script associated with the `download_cursor` event is a `SELECT` statement that defines the rows to be downloaded. The client detects whether the row exists or not and carries out the appropriate insert or update operation.
- ◆ At the end of the download processing, the client automatically deletes rows that violate referential integrity.

☞ For more information, see “Referential integrity and synchronization” [*MobiLink Synchronization User’s Guide*, page 28].

authenticate_parameters connection event

Function	Receives non-table data that is sent from the remote.
Parameters	In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [<i>MobiLink Synchronization User’s Guide</i> , page 233] and “SQL-.NET data types” [<i>MobiLink Synchronization User’s Guide</i> , page 261].

Item	Parameter	Description
1	auth_status	INTEGER. This is an INOUT parameter.
2	ml_username	VARCHAR(128).
3...	remote_parameters (one or more)	VARCHAR(128).

Description You can send strings or parameters in the form of strings from both Adaptive Server Anywhere and UltraLite remotes. This allows you to have authentication beyond a user ID and password. It also means that you can customize your synchronization based on the value of parameters, and do this in a pre-synchronization phase, during authentication.

For UltraLite remote databases, pass the parameters using the num_auth_parms and auth_parms fields in the ul_sync_info struct. num_auth_parms is a count of the number of parameters, from 0 to 255. auth_parms is a pointer to a list of strings. To prevent the strings being viewed as plain text, the strings are sent in the same way as passwords. If num_auth_parms is 0, set auth_parms to NULL.

Following is an example of passing parameters in UltraLite:

```
ul_char * Params[ 3 ] = { UL_TEXT( "parm1" ), UL_TEXT( "parm2"
                        ), UL_TEXT( "parm3" ) };

...
info.num_auth_parms = 3;
info.auth_parms = Params;
```

For Adaptive Server Anywhere remote databases, you pass parameters using the dbmlsync -ap option, in a comma-separated list. For example,

```
dbmlsync -ap "parm1,param2,param3"
```

The MobiLink synchronization server executes this event upon starting each synchronization. It is executed before, and in the same transaction as, the

`begin_synchronization` event.

You can use this event to replace the built-in MobiLink authentication mechanism with a custom mechanism. You may want to call into the authentication mechanism of your DBMS, or you may wish to implement features not present in the MobiLink built-in mechanism.

The number of remote parameters must match the number expected or an error will result. For example, if three parameters are sent, the event must expect five, because there is `auth_status` and `ml_username` plus the three parameters. An error will also occur if parameters are sent from the client and there is no event.

If the `authenticate_user` or `authenticate_user_hashed` scripts are invoked and return an error, this event is not called.

SQL scripts for the `authenticate_parameters` event must be implemented as stored procedures.

See also

“Authenticating MobiLink Users” [*MobiLink Synchronization User’s Guide*, page 103]

“Custom user authentication” [*MobiLink Synchronization User’s Guide*, page 114]

[“authenticate_user connection event” on page 100](#)

[“authenticate_user_hashed connection event” on page 104](#)

[“begin_synchronization connection event” on page 121](#)

“Authentication Parameters synchronization parameter” [*UltraLite Database User’s Guide*, page 162]

authenticate_user connection event

Function Implements a custom user authentication mechanism.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	auth_status	INTEGER. This is an INOUT parameter.
2	ml_username	VARCHAR(128).
3	user_password	VARCHAR(128).
4	user_new_password	VARCHAR(128).

Default action Use MobiLink built-in user authentication mechanism.

Description The MobiLink synchronization server executes this event upon starting each synchronization. It is executed in a transaction before the begin_synchronization transaction.

You can use this event to replace the built-in MobiLink authentication mechanism with a custom mechanism. You may want to call into the authentication mechanism of your DBMS, or you may wish to implement features not present in the MobiLink built-in mechanism, such as password expiry or a minimum password length.

The parameters used in an authenticate_user event are as follows:

1. **auth_status** This required parameter is an INOUT parameter: a parameter that provides a value to the script, and could be given a new value by the script. The auth_status parameter indicates the overall success of the authentication, and can be set to one of the following values:

Returned Value	Auth_status	Description
$V \leq 1999$	1000	Authentication succeeded.
$1999 < V \leq 2999$	2000	Authentication succeeded: password expiring soon.
$2999 < V \leq 3999$	3000	Authentication failed: password expired.
$3999 < V \leq 4999$	4000	Authentication failed.
$4999 < V \leq 5999$	5000	Authentication failed as user is already synchronizing.
$5999 < V$	4000	If the returned value is greater than 5999, MobiLink interprets it as a returned value of 4000.

2. **ml_username** This optional parameter is the name of the remote database.
3. **user_password** This optional parameter indicates the password for authentication purposes. If the user does not supply a password, this is NULL.
4. **user_new_password** This optional parameter indicates a new password. If the user does not change their password, this is NULL.

SQL scripts for the `authenticate_user` event must be implemented as stored procedures.

When the two authentication scripts are both defined, and both scripts return different `auth_status` codes, the higher value is used.

The `authenticate_user` script is executed immediately before, and in the same transaction as, the `begin_synchronization` script. The transaction is ended immediately after the `begin_synchronization` script.

See also

“Authenticating MobiLink Users” [*MobiLink Synchronization User’s Guide*, page 103]

“Custom user authentication” [*MobiLink Synchronization User’s Guide*, page 114]

[“authenticate_user_hashed connection event” on page 104](#)

[“authenticate_parameters connection event” on page 98](#)

[“modify_user connection event” on page 184](#)

[“begin_synchronization connection event” on page 121](#)

SQL example

A typical `authenticate_user` script is a call to a stored procedure. The order of the parameters in the call must match the order above. In an Adaptive Server Anywhere consolidated database, the script could be as follows.

```
call my_auth ( ?, ?, ?, ? )
```

The following Adaptive Server Anywhere stored procedure uses only the user name to authenticate—it has no password check. The procedure ensures only that the supplied user name is one of the employee IDs listed in the `ULEmployee` table.

```
CREATE PROCEDURE my_auth(in @user_name varchar(128))
begin
    if exists
    ( select * from ulemmployee
      where emp_id = @user_name )
    then
        message 'OK' type info to client;
        return 1000;
    else
        message 'Not OK' type info to client;
        return 4000;
    end if
end
```

Java example

The following stored procedure call registers a Java method called `authenticateUser` as the script for the `authenticate_user` event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_connection_script(
    'ver1', 'authenticate_user',
    'ExamplePackage.ExampleClass.authenticateUser' )
```

Following is the sample Java method `authenticateUser`. It calls Java functions that check and, if needed, change the user's password.

```

public String authenticateUser
( ianywhere.ml.script.InOutInteger authStatus,
  String user, String pwd, String newPwd )
throws java.sql.SQLException
{ // in a real authenticate_user handler, we would
  // handle more auth code states
  _curUser = user;
  if( checkPwd( user, pwd ) )
  { // auth successful
    if( newPwd != null )
    { // pwd is being changed
      if( changePwd( user, pwd, newPwd ) )
      { // auth ok and pwd change ok. Use custom code
        authStatus.setValue( 1001 );
      }
    }
    else { // authorization ok but password
      // change failed. Use custom code.
      java.lang.System.err.println( "user: "
        + user + " pwd change failed!" );
      authStatus.setValue( 1002 ); } }
    else { authStatus.setValue( 1000 ); } }
    else { // auth failed
      authStatus.setValue( 4000 ); }
    return( null ); }

```

.NET example

The following stored procedure call registers a .NET method called `AuthUser` as the script for the `authenticate_user` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_dnet_connection_script(
  'ver1', 'authenticate_user',
  'TestScripts.Test.AuthUser'
)

```

Following is the C# signature for the method `AuthUser`.

```

public void AuthUser( ref int authStatus, string user, string
  pwd, string newPwd )

```

☞ For a more detailed example of an `authenticate_user` script written in C# in .NET, see “.NET synchronization example” [*MobiLink Synchronization User's Guide*, page 266].

authenticate_user_hashed connection event

Function Implements a custom user authentication mechanism.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	auth_status	INTEGER. This is an INOUT parameter.
2	ml_username	VARCHAR(128).
3	hashed_user_password	BINARY(20). If the user does not supply a password, this is NULL.
4	hashed_new_password	BINARY(20). If the user does not change their password, this is NULL.

Default action Use MobiLink built-in user authentication mechanism.

Description This event is identical to `authenticate_user` except for the passwords, which are in the same hashed form as those stored in the `ml_user.hashed_password` column. Passing the passwords in hashed form provides increased security.

A one-way hash is used. A one-way hash takes a password and converts it to a byte sequence that is (essentially) unique to each possible password. The one-way hash lets password authentication take place without having to store the actual password in the consolidated database.

When the two authentication scripts are both defined, and both scripts return different `auth_status` codes, the higher value is used.

See also “Authenticating MobiLink Users” [*MobiLink Synchronization User’s Guide*, page 103]

“Custom user authentication” [*MobiLink Synchronization User’s Guide*, page 114]

[“authenticate_user connection event” on page 100](#)

[“authenticate_parameters connection event” on page 98](#)

SQL example

A typical `authenticate_user_hashed` script is a call to a stored procedure. The order of the parameters in the call must match the order above. In an Adaptive Server Anywhere consolidated database, the script could be as follows.

```
call my_auth ( ?, ?, ? )
```

The following Adaptive Server Anywhere stored procedure uses both the user name and password to authenticate. The procedure ensures only that the supplied user name is one of the employee IDs listed in the `ULEmployee` table. The procedure assumes that the `Employee` table has a `binary(20)` column called `hashed_pwd`.

```
CREATE PROCEDURE my_auth(
  inout @auth_status integer,
  in @user_name varchar(128),
  in @hpwd binary(20) )
begin
  if exists
    ( select * from ulemployee
      where emp_id = @user_name
        and hashed_pwd = @hpwd )
  then
    message 'OK' type info to client;
    return 1000;
  else
    message 'Not OK' type info to client;
    return 4000;
  end if
end
```

Java example

The following stored procedure call registers a Java method called `authUserHashed` as the script for the `authenticate_user_hashed` event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_connection_script(
  'ver1', 'authenticate_user_hashed',
  'ExamplePackage.ExampleClass.authUserHashed')
```

Following is the sample Java method `authUserHashed`. It calls Java functions that check and, if needed, change the user's password.

```

public String authUserHashed(
    ianywhere.ml.script.InOutInteger authStatus,
    String user, byte pwd[], byte newPwd[] )
throws java.sql.SQLException
{ // in a real authenticate_user_hashed handler, we
  // would handle more auth code states
  _curUser = user;
  if( checkPwdHashed( user, pwd ) ) {
    // auth successful
    if( newPwd != null )
    { // pwd is being changed
      if( changePwdHashed( user, pwd, newPwd ) )
      { // auth ok and pwd change ok use custom code
        authStatus.setValue( 1001 ); }
      else
      { // auth ok but pwd change failed.
        // Use custom code
        java.lang.System.err.println( "user: " + user
          + " pwd change failed!" );
        authStatus.setValue( 1002 ); } }
    else { authStatus.setValue( 1000 ); } }
    else { // auth failed
      authStatus.setValue( 4000 ); }
    return( null ); }
}

```

.NET example

The following stored procedure call registers a .NET method called `AuthUserHashed` as the script for the `authenticate_user_hashed` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_dnet_connection_script(
    'ver1',
    'authenticate_user_hashed',
    'TestScripts.Test.AuthUserHashed'
)

```

Following is the C# signature for the call `AuthUserHashed`.

```

public void AuthUserHashed(
    ref int authStatus,
    string user,
    byte[] pwd,
    byte[] newPwd )

```


begin_connection connection event

Function	Invoked at the time the MobiLink synchronization server connects to the consolidated database server.
Parameters	None.
Default action	None.
Description	The MobiLink synchronization server executes this event upon opening each worker-thread connection to the consolidated database server. The MobiLink synchronization opens connections on demand as synchronization requests come in. When an application forms or reforms a connection with the MobiLink synchronization server, the MobiLink synchronization server temporarily allocates one connection with the database server for the duration of that synchronization.
See also	“end_connection connection event” on page 145
SQL example	<p>The following SQL script works in an Adaptive Server Anywhere database. Two variables are created, one for the last_download timestamp, and one for employee ID.</p> <pre> call ml_add_connection_script('custdb', 'begin_connection', 'create variable @LastDownload timestamp; create variable @EmployeeID integer;') </pre>
Java example	<p><i>Note:</i> This script is not generally used in Java, because instead of database variables you would use member variables in this class instance, and prepare the members in the constructor.</p> <p>The following stored procedure call registers a Java method called beginConnection as the script for the begin_connection event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.</p> <pre> call ml_add_java_connection_script('ver1', 'begin_connection', 'ExamplePackage.ExampleClass.beginConnection') </pre> <p>Following is the sample Java method beginConnection. This returns SQL that will create a connection level variable.</p> <pre> public String beginConnection() { return("create variable @LastDownload timestamp;"); } </pre>
.NET example	The following stored procedure call registers a .NET method called

BeginConnection as the script for the begin_connection connection event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_connection_script(  
    'ver1',  
    'begin_connection',  
    'TestScripts.Test.BeginConnection'  
)
```

Following is the signature for the call BeginConnection.

```
public void BeginConnection()
```

begin_connection_autocommit connection event

Function	Turns on autocommit.
Parameters	None.
Default action	Autocommit is off.
Description	When the MobiLink synchronization server connects to the consolidated database, it turns off autocommit so that it can roll back the upload and download streams if an error occurs.

However, if you are using an Adaptive Server Enterprise consolidated database, you cannot perform DDL functions such as creating temporary tables unless autocommit is on. If you are using an Adaptive Server Enterprise consolidated database, run your DDL commands in the `begin_connection_autocommit` event. When the event is finished, autocommit is turned off.

`Begin_connection_autocommit` scripts must be written so that they are repeatable. This is because if an error or deadlock occurs, the MobiLink synchronization server needs to be able to retry the script (since it can't roll it back).

begin_download connection event

Function Processes any statements just before the MobiLink synchronization server commences preparing the download data stream.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. You must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	last_download	TIMESTAMP
2	ml_username	VARCHAR(128)

Default action None.

Description The MobiLink synchronization server executes this event as the first step in the processing of downloaded information. Download information is processed in a single transaction. The execution of this event is the first action in this transaction.

The last_download timestamp is the value obtained from the consolidated database during the last successful synchronization immediately prior to the download phase. If the current user has never synchronized successfully, this value is set to 1900-01-01.

See also [“end_download connection event” on page 147](#)

SQL example The following example works in an Adaptive Server Anywhere installation.

```
call ml_add_connection_script (
    'Lab',
    'begin_download',
    'CALL SetDownloadParameters ( ?, ? )' )
```

Java example The following stored procedure call registers a Java method called beginDownloadConnection as the script for the begin_download connection event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_connection_script(
    'example_ver',
    'begin_download',
    'ExamplePackage.ExampleClass.beginDownloadConnection' )
```

Following is the sample Java method `beginDownloadConnection`. It calls a Java function that will prepare the delete tables using a JDBC synchronization that was set earlier.

```
public String beginDownloadConnection(
    Timestamp ts, String user )
    throws java.sql.SQLException
{   prepDeleteTables ( _syncConn, ts, user );
    return ( null ); }
```

.NET example

The following stored procedure call registers a .NET method called `BeginDownload` as the script for the `begin_download` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_connection_script(
    'ver1',
    'begin_download',
    'TestScripts.Test.BeginDownload'
)
```

Following is the C# signature for the call `BeginDownload`.

```
public void BeginDownload(
    DateTime timestamp,
    string user )
```

begin_download table event

- Function

Provides a location to process statements related to a specific table just before preparing the download stream of inserts, updates, and deletions.
- Parameters

In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [MobiLink Synchronization User’s Guide, page 233] and “SQL-.NET data types” [MobiLink Synchronization User’s Guide, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	last_download	TIMESTAMP
2	ml_username	VARCHAR(128)
3	table	VARCHAR (128)

- Default action

None.
- Description

The MobiLink synchronization server executes this event as the first step in preparing download information for a specific table. The download information is prepared in its own transaction. The execution of this event is the first table-specific action in the transaction.

You can have one begin_download script for each table in the remote database. The script is only invoked when that table is synchronized.

The last_download timestamp is the value obtained from the consolidated database during the last successful synchronization immediately prior to the download phase. If the current user has never synchronized successfully, this value is set to 1900-01-01.
- See also

[“end_download table event” on page 149](#)
- SQL example

The following example can be used on an Adaptive Server Anywhere 9 database. The first chunk of code calls the ml_add_table_script, and the second creates a BeginTableDownload procedure.

```

call ml_add_table_script(
    'version1',
    'Leads',
    'begin_download',
    'call BeginTableDownload( ?, ?, ? ) );
create procedure BeginTableDownload(
    LastDownload timestamp,
    MLUser varchar(128),
    TableName varchar(128) )
begin
    execute immediate 'update ' || TableName ||
' set last_download_check = CURRENT_TIMESTAMP
where Owner = ' || MLUser;
end

```

Java example

The following stored procedure call registers a Java method called `beginDownloadTable` as the script for the `begin_download` table event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_java_table_script(
    'ver1',
    'table1',
    'begin_download',
    'ExamplePackage.ExampleClass.beginDownloadTable' )

```

Following is the sample Java method `beginDownloadTable`. It saves the name of the current table for use in a later member function call.

```

public String beginDownloadTable(
    Timestamp ts,
    String user,
    String table )
{
    _curTable = table;
    return( null );
}

```

.NET example

The following stored procedure call registers a .NET method called `BeginTableDownload` as the script for the `begin_download` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_dnet_table_script(
    'ver1', 'table1', 'begin_download',
    'TestScripts.Test.BeginTableDownload'
)

```

Following is the C# signature for the call `BeginTableDownload`.

```

public void BeginTableDownload(
    DateTime timestamp,
    string user,
    string table )

```

begin_download_deletes table event

Function Processes statements related to a specific table just before fetching a list of rows to be deleted from the specified table in the remote database.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [MobiLink Synchronization User’s Guide, page 233] and “SQL-.NET data types” [MobiLink Synchronization User’s Guide, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	last_download	TIMESTAMP
2	ml_username	VARCHAR (128)
3	table	VARCHAR (128)

Default action None.

Description This event is executed immediately before fetching a list of rows to be deleted from the named table in the remote database.

You can have one begin_download_deletes script for each table in the remote database.

The last_download timestamp is the value obtained from the consolidated database during the last successful synchronization immediately prior to the download phase. If the current user has never synchronized successfully, this value is set to 1900-01-01.

See also [“begin_download_rows table event” on page 116](#)
[“end_download_rows table event” on page 153](#)

SQL example To minimize the amount of data on remotes, you can use this event to flag data that will be deleted when the download_delete_cursor is executed. The following example flags for deletion sales leads from the remote device that are over 10 weeks old. The example can be used on an Adaptive Server Anywhere 9 database. The code calls the ml_add_table_script, and then creates a beginDownloadDeletes procedure.


```

call ml_add_table_script (
    'version1',
    'Leads',
    'begin_download_deletes',
    'call BeginDownloadDeletes (?, ?, ?)' );
create procedure BeginDownloadDeletes(
    LastDownload timestamp,
    MLUser varchar(128),
    TableName varchar(128) )
begin
    execute immediate 'update ' || TableName ||
    ' set delete_flag = 1 where
    days(creation_time, CURRENT DATE) > 70 and Owner = '
    || MLUser;
end;

```

Java example

The following stored procedure call registers a Java method called `beginDownloadDeletes` as the script for the `begin_download_deletes` table event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_java_table_script(
    'ver1',
    'table1',
    'begin_download_deletes',
    'ExamplePackage.ExampleClass.beginDownloadDeletes' )

```

Following is the sample Java method `beginDownloadDeletes`. It saves the name of the current table for use in a later member function call.

```

public String beginDownloadDeletes( Timestamp ts,
String user, String table )
{
    _curTable = table;
    return( null ); }

```

.NET example

The following stored procedure call registers a .NET method called `BeginDownloadDeletes` as the script for the `begin_download_deletes` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_dnet_table_script(
    'ver1', 'table1', 'begin_download_deletes',
    'TestScripts.Test.BeginDownloadDeletes'
)

```

Following is the C# signature for the call `BeginDownloadDeletes`.

```

public void BeginDownloadDeletes(
    DateTime timestamp,
    string user,
    string table )

```

begin_download_rows table event

Function Processes statements related to a specific table just before fetching a list of rows to be inserted or updated in the specified table in the remote database.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [MobiLink Synchronization User’s Guide, page 233] and “SQL-.NET data types” [MobiLink Synchronization User’s Guide, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	last_download	TIMESTAMP
2	ml_username	VARCHAR (128)
3	table	VARCHAR (128)

Default action None.

Description This event is executed immediately before fetching the stream of rows to be inserted or updated in the named table in the remote database.

You can have one begin_download_rows script for each table in the remote database.

The last_download timestamp is the value obtained from the consolidated database during the last successful synchronization immediately prior to the download phase. If the current user has never synchronized successfully, this value is set to 1900-01-01.

See also [“begin_download_deletes table event” on page 114](#)
[“end_download_deletes table event” on page 151](#)

SQL example You can use the begin_download_rows table event to flag rows that you no longer want to download for this table. The following example archives sales leads that are over seven days old.

```

call ml_add_table_script( 'version1', 'Leads',
    'begin_download_rows',
    'call BeginDownloadRows (?, ?, ?)' );
create procedure BeginDownloadRows (
    LastDownload timestamp, MLUser varchar(128),
    TableName varchar(128) )
begin
    execute immediate 'update ' || TableName ||
        ' set download_flag = 0 where
        days(creation_time, CURRENT DATE) > 7 and Owner = '
        || MLUser;
end;

```

Java example

The following stored procedure call registers a Java method called `beginDownloadRows` as the script for the `begin_download_rows` table event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_java_table_script(
    'ver1',
    'table1',
    'begin_download_rows',
    'ExamplePackage.ExampleClass.beginDownloadRows' )

```

Following is the sample Java method `beginDownloadRows`. It generates an UPDATE statement using the table and user. MobiLink will execute this SQL statement.

```

public String beginDownloadRows( Timestamp ts,
    String user, String table )
{ return( "update " + table + " set download_flag = 0 "
    + " where days(creation_time, CURRENT DATE) > 7 " +
    " and Owner = '" + user + "'" ); }

```

.NET example

The following stored procedure call registers a .NET method called `BeginDownloadRows` as the script for the `begin_download_rows` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_dnet_table_script(
    'ver1', 'table1', 'begin_download_rows',
    'TestScripts.Test.BeginDownloadRows'
)

```

Following is the C# signature for the call `BeginDownloadRows`.

```

public void BeginDownloadRows(
    DateTime timestamp,
    string user,
    string table )

```

begin_publication connection event

- Function

Provides useful information about the publication(s) being synchronized. This script may also be used to manage generation numbers for file-based downloads.
- Parameters

In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [MobiLink Synchronization User’s Guide, page 233] and “SQL-.NET data types” [MobiLink Synchronization User’s Guide, page 261].

Item	Parameter	Description
1	generation_number	INTEGER. This is an INOUT parameter. If your deployment does not use file-based downloads, this parameter can be ignored. The default is 1.
2	ml_username	VARCHAR(128). If an UltraLite remote is synchronizing with UL_SYNC_ALL, this event is invoked once with the name ‘unknown’.
3	publication_name	VARCHAR(128)
4	last_upload	TIMESTAMP. Last successful upload.
5	last_download	TIMESTAMP. Last successful download.

- Default action

The default generation number is 1. If no script is defined for this event, the generation number sent to the remote will always be 1.
- Description

This event lets you design synchronization logic based on the publications currently being synchronized. This event is invoked in the same transaction as the begin_synchronization event, and is invoked after the begin_synchronization event. It is invoked once per publication being synchronized.

One potential use for this event is to affect what is downloaded based on the publication used. For example, consider a table that is part of both a priority publication (PriorityPub) and a publication for all tables (AllTablesPub). A script for the begin_publication event could store the publication names in a Java class or a SQL variable or package. Download scripts could then behave differently based on whether the publication being synchronized is

	PriorityPub or AllTablesPub.
Generation number	<p>The <code>generation_number</code> parameter is specifically for file-based downloads. The output value of the generation number is passed from the <code>begin_synchronization</code> script to the <code>end_synchronization</code> script. The meaning of the <code>generation_number</code> depends on whether the current synchronization is being used to create a download file, or whether the current synchronization has an upload.</p> <p>The output value of the generation number is passed from the <code>begin_publication</code> script to the <code>end_publication</code> script. The meaning of the <code>generation_number</code> depends on whether the current synchronization is being used to create a download file, or whether the current synchronization has an upload.</p> <p>In file-based downloads, generation numbers are used to force an upload before the download. The number is stored in the download file. During a synchronization that has an upload, one generation number is output for every subscription to a publication. They are sent to the remote database in the upload acknowledgement, and stored in <code>SYSSYNC.generation_number</code>.</p>
See also	<p>“end_publication connection event” on page 155</p> <p>“File-Based Downloads” [<i>MobiLink Synchronization User’s Guide</i>, page 117]</p>
SQL example	<p>You may want to record the information for each publication being synchronized:</p> <pre>call ml_add_connection_script('version1', 'begin_publication', '{call RecordPubSync(?, ?, ?, ?, ?)}');</pre>
Java example	<p>The following stored procedure call registers a Java method called <code>beginPublication</code> as the script for the <code>begin_publication</code> connection event when synchronizing the script version <code>ver1</code>. This syntax is for Adaptive Server Anywhere consolidated databases.</p> <pre>call ml_add_java_connection_script('ver1', 'begin_publication', 'ExamplePackage.ExampleClass.beginPublication')</pre> <p>Following is the sample Java method <code>beginPublication</code>. It saves the name of each publication for later use.</p>

```
public String beginPublication(
    ianywhere.ml.script.InOutInteger generation_number,
    String user,
    String pub_name,
    Timestamp last_upload,
    Timestamp last_download )
{
    _publicationNames[ _numPublications++ ] = pub_name;
    return( null );
}
```

.NET example

The following stored procedure call registers a .NET method called **BeginPub** as the script for the `begin_publication` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_connection_script( 'ver1',
    'begin_publication',
    'TestScripts.Test.BeginPub'
)
```

Following is the C# signature for **BeginPub**.

```
public void BeginPub(
    ref int generation_number,
    string user,
    string pub_name,
    DateTime last_upload,
    DateTime last_download )
{
    _publicationNames[ _numPublications++ ] = pub_name;
}
```

begin_synchronization connection event

Function	Processes any statements at the time an application connects to the MobiLink synchronization server in preparation for the synchronization process.
Parameters	In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [<i>MobiLink Synchronization User’s Guide</i> , page 233] and “SQL-.NET data types” [<i>MobiLink Synchronization User’s Guide</i> , page 261].

Item	Parameter	Description
1	ml_username	VARCHAR(128)

Default action None.

Description The MobiLink synchronization server executes this event immediately after an application preparing to synchronize has formed a connection with the MobiLink synchronization server.

This event is executed within a separate transaction before the upload transaction. It is useful for maintaining statistics.

See also [“end_synchronization connection event” on page 158](#)
[“begin_synchronization table event” on page 123](#)

SQL example You may want to store the ml_username value in a temporary table or variable if you will be referencing that value many times in subsequent scripts.

```
Call ml_add_connection_script (
    'version1',
    'begin_synchronization',
    'set @EmployeeID = ?' );
```

Java example The following stored procedure call registers a Java method called beginSynchronizationConnection as the script for the begin_synchronization connection event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_connection_script( 'ver1',
    'begin_synchronization',
    'ExamplePackage.ExampleClass.beginSynchronizationConnection'
)
```

Following is the sample Java method beginSynchronizationConnection. It saves the name of the synchronizing user for later use.

```
public String beginSynchronizationConnection(  
    String user )  
{  
    _curUser = user;  
    return( null ); }  
}
```

.NET example

The following stored procedure call registers a .NET method called BeginSync as the script for the begin_synchronization connection event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_connection_script( 'ver1',  
    'begin_synchronization',  
    'TestScripts.Test.BeginSync'  
)
```

Following is the C# signature for the call BeginSync.

```
public void BeginSync( string user )
```


begin_synchronization table event

Function	Processes statements related to a specific table at the time an application connects to the MobiLink synchronization server in preparation for the synchronization process.										
Parameters	<p>In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [<i>MobiLink Synchronization User’s Guide</i>, page 233] and “SQL-.NET data types” [<i>MobiLink Synchronization User’s Guide</i>, page 261].</p> <p>Event parameters are optional only if no subsequent parameters are specified. You must use parameter 1 if you want to use parameter 2.</p> <table> <tr> <th>Item</th><th>Parameter</th><th>Description</th></tr> <tr> <td>1</td><td>ml_username</td><td>VARCHAR (128)</td></tr> <tr> <td>2</td><td>table</td><td>VARCHAR (128)</td></tr> </table>		Item	Parameter	Description	1	ml_username	VARCHAR (128)	2	table	VARCHAR (128)
Item	Parameter	Description									
1	ml_username	VARCHAR (128)									
2	table	VARCHAR (128)									
Default action	None.										
Description	<p>The MobiLink synchronization server executes this event after an application that is preparing to synchronize has formed a connection with the MobiLink synchronization server, and after the begin_synchronization connection-level event.</p> <p>You can have one begin_synchronization script for each table in the remote database. The event is only invoked when the table is synchronized.</p>										
See also	<p>“end_synchronization table event” on page 160</p> <p>“begin_synchronization connection event” on page 121</p>										
SQL example	The begin_synchronization table event is used to set up the synchronization of a particular table. The following Adaptive Server Anywhere SQL procedure call registers a script that creates a temporary table for storing rows during synchronization.										

```

call ml_add_table_script(
  'ver1',
  'sales_order',
  'begin_synchronization',
  'CREATE TABLE #sales_order
(
  id            integer NOT NULL default autoincrement,
  cust_id       integer NOT NULL,
  order_date    date NOT NULL,
  fin_code_id   char(2) NULL,
  region        char(7) NULL,
  sales_rep     integer NOT NULL,
  PRIMARY KEY (id),
)' )

```

Java example

The following stored procedure call registers a Java method called `beginSynchronizationTable` as the script for the `begin_synchronization` table event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_java_table_script(
  'ver1',
  'table1',
  'begin_synchronization',
  'ExamplePackage.ExampleClass.beginSynchronizationTable' )

```

Following is the sample Java method `beginSynchronizationTable`. It adds the current table name to a list of table names contained in this instance.

```

public String beginSynchronizationTable(String user,
String table )
{
  _tableList.add( table );
  return( null ); }

```

.NET example

The following stored procedure call registers a .NET method called `BeginTableSync` as the script for the `begin_synchronization` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_dnet_table_script(
  'ver1', 'table1',
  'begin_synchronization',
  'TestScripts.Test.BeginTableSync'
)

```

Following is the C# signature for the call `BeginTableSync`.

```

public void BeginTableSync( string user, string table )

```

begin_upload connection event

Function	Processes any statements just before the MobiLink synchronization server commences processing the stream of uploaded inserts, updates, and deletes.
Parameters	In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [<i>MobiLink Synchronization User’s Guide</i> , page 233] and “SQL-.NET data types” [<i>MobiLink Synchronization User’s Guide</i> , page 261].

Item	Parameter	Description
1	ml_username	VARCHAR (128)

Default action	None.
Description	The MobiLink synchronization server executes this event as the first step in the processing of uploaded information. Upload information is processed in a single transaction. The execution of this event is the first action in this transaction.
See also	“end_upload connection event” on page 162 “begin_upload table event” on page 127
SQL example	The begin_upload connection event is used to perform whatever steps you need performed prior to uploading rows. The following Adaptive Server Anywhere SQL script creates a temporary table for storing old and new row values for conflict processing of the sales_order table.

```
call ml_add_connection_script(
  'version1',
  'begin_upload',
  'CREATE TABLE #sales_order_conflicts
  (
    id          integer NOT NULL default autoincrement,
    cust_id     integer NOT NULL,
    order_date  date NOT NULL,
    fin_code_id char(2) NULL,
    region     char(7) NULL,
    sales_rep   integer NOT NULL,
    new_value   char(1) NOT NULL,
    PRIMARY KEY (id),
  )' )
```

Java example	The following stored procedure call registers a Java method called beginUploadConnection as the script for the begin_upload connection event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.
--------------	--

```
call ml_add_java_connection_script( 'ver1', 'begin_upload',
'ExamplePackage.ExampleClass. beginUploadConnection ' )
```

Following is the sample Java method `beginUploadConnection`. It prints a message to the MobiLink output log. (This might be useful at development time but would slow down a production server.)

```
public String beginUploadConnection( String user )
{  java.lang.System.out.println(
    "Starting upload for user: " + user );
  return( null ); }
```

.NET example

The following stored procedure call registers a .NET method called `BeginUpload` as the script for the `begin_upload` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_connection_script(
'ver1',
'begin_upload',
'TestScripts.Test.BeginUpload'
)
```

Following is the C# signature for the call `BeginUpload`.

```
public void BeginUpload( string user )
```

The following C# example saves the current user name for use in a later event.

```
public void BeginUpload( string curUser )
{
    user = curUser;
}
```

begin_upload table event

Function	Processes statements related to a specific table just before the MobiLink synchronization server commences processing the stream of uploaded inserts, updates, and deletes.
Parameters	<p>In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [<i>MobiLink Synchronization User’s Guide</i>, page 233] and “SQL-.NET data types” [<i>MobiLink Synchronization User’s Guide</i>, page 261].</p> <p>Event parameters are optional only if no subsequent parameters are specified. You must use parameter 1 if you want to use parameter 2.</p>

Item	Parameter	Description
1	ml_username	VARCHAR(128)
2	table	VARCHAR(128)

Default action	None.
Description	<p>The MobiLink synchronization server executes this event as the first step in the processing of uploaded information. Upload information is processed in a separate transaction. The execution of this event is the first table-specific action in this transaction.</p> <p>You can have one begin_upload script for each table in the remote database. The script is only invoked when the table is actually synchronized.</p>
See also	<p>“end_upload table event” on page 164</p> <p>“begin_upload connection event” on page 125</p>
SQL example	<p>When uploading rows from a remote you may want to place the changes in an intermediate table and manually process changes yourself. You can populate a global temporary table in this event.</p>

```
call ml_add_table_script(
    'version1',
    'Leads',
    'begin_upload',
    'insert into T_Leads SELECT *
FROM Leads WHERE Owner = @EmployeeID')
```

Java example	The following stored procedure call registers a Java method called beginUploadTable as the script for the begin_upload table event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.
--------------	--

```
call ml_add_java_table_script(  
    'ver1',  
    'table1',  
    'begin_upload',  
    'ExamplePackage.ExampleClass.a beginUploadTable'  
)
```

Following is the sample Java method `beginUploadTable`. This example takes no action. MobiLink interprets NULL as no script.

```
public String beginUploadTable( String user,  
    String table )  
{    return( null ); }
```

.NET example

The following stored procedure call registers a .NET method called `BeginTableUpload` as the script for the `begin_upload` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script(  
    'ver1', 'table1', 'begin_upload',  
    'TestScripts.Test.BeginTableUpload'  
)
```

Following is the C# signature for the call `BeginTableUpload`.

```
public void BeginTableUpload(  
    string user,  
    string table )
```

begin_upload_deletes table event

- Function** Processes statements related to a specific table just before uploading deleted rows from the specified table in the remote database.
- Parameters** In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. You must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	ml_username	VARCHAR(128)
2	table	VARCHAR(128)

- Default action** None.

- Description** This event runs immediately before applying the changes that result from rows deleted in the client table named in the second parameter.
- You can have one begin_upload_deletes script for each table in the remote database. The script is only invoked when the table is actually synchronized.

- See also** [“end_upload_deletes table event” on page 166](#)

- SQL example** The begin_upload_deletes connection event is used to perform whatever steps you need performed after uploading inserts and updates for a particular table, but before deletes are uploaded for that table. The following Adaptive Server Anywhere SQL script creates a temporary table for storing deletes temporarily during upload:

```
call ml_add_table_script(
  'ver1',
  'sales_order',
  'begin_upload_deletes',
  'CREATE TABLE #sales_order_deletes
(
  id          integer NOT NULL default autoincrement,
  cust_id     integer NOT NULL,
  order_date  date NOT NULL,
  fin_code_id char(2) NULL,
  region     char(7) NULL,
  sales_rep   integer NOT NULL,
  PRIMARY KEY (id),
)' )
```

Java example

The following stored procedure call registers a Java method called `beginUploadDeletes` as the script for the `begin_upload_deletes` table event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script( 'ver1', 'table1',  
    'begin_upload_deletes',  
    'ExamplePackage.ExampleClass. beginUploadDeletes' )
```

Following is the sample Java method `beginUploadDeletes`. It prints a message to the MobiLink output log. (This might be useful at development time but would slow down a production server.)

```
public String beginUploadDeletes(  
    String user,  
    String table )  
    throws java.sql.SQLException  
{ java.lang.System.out.println( "Starting upload  
    deleted for table: " + table );  
    return( null ); }
```

.NET example

The following stored procedure call registers a .NET method called `BeginUploadDeletes` as the script for the `begin_upload_deletes` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script( 'ver1', 'table1',  
    'begin_upload_deletes',  
    'TestScripts.Test.BeginUploadDeletes'  
    )
```

Following is the C# signature for the call `BeginUploadDeletes`.

```
public void BeginUploadDeletes( string user,  
    string table )
```


begin_upload_rows table event

- Function** Processes statements related to a specific table just before uploading inserts and updates from the specified table in the remote database.
- Parameters** In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. You must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	ml_username	VARCHAR(128)
2	table	VARCHAR(128)

- Default action** None.

- Description** This event is run immediately prior to applying the changes that result from inserts and deletes to the client table named in the second parameter.
- You can have one begin_upload_rows script for each table in the remote database. The script is only invoked when the table is actually synchronized.

- See also** [“end_upload_rows table event” on page 168](#)

- SQL example** The begin_upload_rows connection event is used to perform whatever steps you need performed before uploading inserts and updates for a particular table. The following Adaptive Server Anywhere SQL script calls a stored procedure that prepares the consolidated database for inserts and updates into the Inventory table:

```
call ml_add_table_script(
    'MyCorp 1.0',
    'Inventory',
    'begin_upload_rows',
    'call PrepareForUpserts()' )
```

- Java example** The following stored procedure call registers a Java method called beginUploadRows as the script for the begin_upload_rows table event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(
    'ver1',
    'table1',
    'begin_upload_rows',
    'ExamplePackage.ExampleClass.beginUploadRows' )
```

Following is the sample Java method `beginUploadRows`. It prints a message to the MobiLink output log. (This might be useful at development time but would slow down a production server.)

```
public String beginUploadRows( String user,
String table )
    throws java.sql.SQLException
{   java.lang.System.out.println( "Starting upload rows
    for table: " + table + " and user: " + user );
    return( null ); }
```

.NET example

The following stored procedure call registers a .NET method called `BeginUploadRows` as the script for the `begin_upload_rows` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script(
    'ver1', 'table1', 'begin_upload_rows',
    'TestScripts.Test.BeginUploadRows'
)
```

Following is the C# signature for the call `BeginUploadRows`.

```
public void BeginUploadRows(
    string user,
    string table )
```

download_cursor cursor event

Function Defines a cursor to select rows that are to be downloaded and inserted or updated in the remote database.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. You must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	last_download	TIMESTAMP
2	ml_username	VARCHAR(128)

Default action None.

A default download_cursor SQL script can be generated using the MobiLink synchronization server -za option. Also, the UltraLite analyzer generates a SELECT statement based on your reference database that you can use to get started.

Description The MobiLink synchronization server opens a read-only cursor with which to fetch a list of rows to download to the remote database. This script should contain a suitable SELECT statement.

The parameters are the last_download timestamp and the user name. You can use these values if you choose by placing question marks in your SQL statement.

You can have one download_cursor script for each table in the remote database.

The last_download timestamp is the value obtained from the consolidated database during the last successful synchronization immediately prior to the download phase. If the current user has never synchronized successfully, this value is set to 1900-01-01.

To optimize performance of the download stage of synchronization to UltraLite clients, when the range of primary key values is outside the current rows on the device, you should order the rows in the download cursor by primary key. Downloads of large reference tables, for example, can benefit from this optimization.

Note that `download_cursor` allows for cascading deletes. Thus, you can delete records from a database.

For Java and .NET applications, this script must return valid SQL.

See also

[“download_delete_cursor cursor event” on page 136](#)

SQL example

The following example comes from an Oracle installation, although the statement is valid against all supported databases. The example downloads all rows that have been changed since the last time the user downloaded data, and which match the user name in the `emp_name` column.

```
call ml_add_table_script(
  'Lab',
  'ULOrder',
  'download_cursor',
  'SELECT order_id, cust_id, prod_id, emp_id,
    disc, quant, notes, status
  FROM ULOrder
  WHERE last_modified >= ? AND emp_name = ?' )
```

To write a `download_cursor` SQL script that does not use the first parameter (the `last_download` timestamp), but does use the second parameter (the MobiLink user name), add a dummy clause that affects no rows. For example:

```
call ml_add_table_script(
  'Lab',
  'ULOrder',
  'download_cursor',
  'SELECT order_id, cust_id, prod_id, emp_id, disc,
    quant, notes, status
  FROM ULOrder WHERE ? IS NOT NULL AND emp_name = ?' )
```

You must still use both parameters, but the first `?` is a place holder that does nothing.

Java example

The following stored procedure call registers a Java method called `downloadCursor` as the script for the `download_cursor` cursor event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(
  'ver1',
  'table1',
  'download_cursor',
  'ExamplePackage.ExampleClass.downloadCursor ' )
```

Following is the sample Java method `downloadCursor`. It dynamically creates the SQL statement for the download cursor.

```

public String downloadCursor( Timestamp ts,
String user )
{ return( "SELECT order_id, cust_id, prod_id, emp_id,
disc, " + " quant, notes, status " + "FROM ULOrder " +
"WHERE emp_name = '" + user + "'" ); }

```

.NET example

The following stored procedure call registers a .NET method called DownloadCursor as the script for the download_cursor cursor event when synchronizing the script version ver1 and the table table1. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_dnet_table_script(
'ver1', 'table1', 'download_cursor',
'TestScripts.Test.DownloadCursor'
)

```

Following is the C# signature for the call DownloadCursor.

```

public string DownloadCursor(
    DateTime timestamp,
    string user )

```

The following C# example populates a temporary table with the contents of a file called rows.txt. It then returns a cursor that causes MobiLink to send the rows in the temporary table to the remote database.

```

public string DownloadCursor(
    DateTime ts,
    string user )
{
    DBCommand stmt = curConn.CreateCommand();
    StreamReader input = new StreamReader( "rows.txt" );
    string sql = input.ReadLine();

    stmt.CommandText = "DELETE FROM dnet_dl_temp";
    stmt.ExecuteNonQuery();

    while( sql != null ){
        stmt.CommandText = "INSERT INTO dnet_dl_temp VALUES " + sql;
        stmt.ExecuteNonQuery();
        sql = input.ReadLine();
    }
    return( "SELECT * FROM dnet_dl_temp" );
}

```

download_delete_cursor cursor event

Function Defines a cursor to select rows that are to be deleted in the remote database.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. You must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	last_download	TIMESTAMP
2	ml_username	VARCHAR(128)

Default action None.

Description The MobiLink synchronization server opens a read-only cursor with which to fetch a list of rows to download, and then insert or update in the remote database. This script must contain a SELECT statement that returns the primary key values of the rows to be deleted from the table in the remote database.

The parameters are the last_download timestamp and the user name. You can use these values by placing a question mark in your SQL statement.

You can have one download_delete_cursor script for each table in the remote database.

If the download_delete_cursor has NULLs for the primary key columns for one or more rows in a table, then MobiLink tells the remote to delete all the data in the table. For a complete description of this behavior, see “Deleting all the rows in a table” [*MobiLink Synchronization User’s Guide*, page 59].

The last_download timestamp is the value obtained from the consolidated database during the last successful synchronization immediately prior to the download phase. If the current user has never synchronized successfully, this value is set to 1900-01-01.

Note that rows deleted from the consolidated database will not appear in a result set defined by a download_delete_cursor event, and so are not automatically deleted from the remote database. One technique for identifying rows to be deleted from remote databases is to add a column to the consolidated database table identifying a row as inactive.

For Java and .NET applications, this script must return valid SQL.

See also

[“download_cursor cursor event” on page 133](#)

SQL example

This example is taken from the Contact sample and can be found in *Samples\MobiLink\Contact\build_consol.sql*. It deletes from the remote database any customers that:

- ◆ have been changed since the last time this user downloaded data (Customer.last_modified > ?), and either
- ◆ do not belong to the synchronizing user (SalesRep.ml_username != ?), or
- ◆ are marked as inactive in the consolidated database (Customer.active = 0).

```
SELECT cust_id FROM Customer key join SalesRep
WHERE Customer.last_modified > ? AND
( SalesRep.ml_username != ? OR Customer.active = 0 )
```

Java example

The following stored procedure call registers a Java method called `downloadDeleteCursor` as the script for the `download_delete_cursor` event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(
    'ver1',
    'table1',
    'download_delete_cursor',
    'ExamplePackage.ExampleClass.downloadDeleteCursor' )
```

Following is the sample Java method `downloadDeleteCursor`. It calls a Java method that generates the SQL for the download delete cursor.

```
public String downloadDeleteCursor( Timestamp ts,
    String user )
{ return( getDownloadCursor( _curUser, _curTable ) ); }
```

.NET example

The following stored procedure call registers a .NET method called `DownloadDeleteCursor` as the script for the `download_delete_cursor` event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script(
    'ver1',
    'table1',
    'download_delete_cursor',
    'TestScripts.Test.DownloadDeleteCursor'
)
```

Following is the C# signature for the call `DownloadDeleteCursor`.

```
public string DownloadDeleteCursor(  
    DateTime timestamp,  
    string user )
```


download_statistics connection event

Function Tracks synchronization statistics for download operations.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	ml_username	VARCHAR(128). The MobiLink user name as specified in your SYNCHRO-NIZATION USER definition.
2	warnings	INTEGER. The number of warnings issued.
3	errors	INTEGER. The number of errors, including handled errors, that occurred.
4	fetches_rows	INTEGER. The number of rows fetched by the download_cursor script.
5	deleted_rows	INTEGER. The number of rows fetched by the download_deletes script.
6	filtered_rows	INTEGER. The number of rows from (5) actually sent to the remote. This reflects download filtering of uploaded values.
7	bytes	INTEGER. The number of bytes sent to the remote as the download.

Default action None.

Description The download_statistics event allows you to gather, for any user, statistics on downloads. The download_statistics connection script is called just prior to the commit at the end of the download transaction.

Note:

Depending on the command line, not all warnings or errors are logged, so the warnings and errors counts may be more than the number of warnings or errors logged.

See also

[“download_statistics table event” on page 142](#)

[“upload_statistics connection event” on page 224](#)

[“upload_statistics table event” on page 227](#)

[“synchronization_statistics connection event” on page 202](#)

[“synchronization_statistics table event” on page 205](#)

[“time_statistics connection event” on page 207](#)

[“time_statistics table event” on page 209](#)

[“MobiLink Monitor” \[MobiLink Synchronization User’s Guide, page 297\]](#)

SQL example

The following example comes from an Oracle installation.

```
INSERT INTO download_audit (id, user_name, warnings,
errors, deleted_rows, fetched_rows, download_rows, bytes)
VALUES (d_audit.nextval, ?,?,?,?,?,,?)
```

Once vital statistics are inserted into the audit table, you may use these statistics to monitor your synchronizations and make optimizations where applicable.

Java example

The following stored procedure call registers a Java method called `downloadStatisticsConnection` as the script for the `download_statistics` event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_connection_script( 'ver1', 'download_
statistics',
'ExamplePackage.ExampleClass.downloadStatisticsConnecti
on' )
```

Following is the sample Java method `downloadStatisticsConnection`. It prints the number of fetched rows to the MobiLink output log.

```
public String downloadStatisticsConnection(
    String user,
    int warnings,
    int errors,
    int fetchedRows,
    int deletedRows,
    int bytes )
{
    java.lang.System.out.println( "download connection
stats fetchedRows: " + fetchedRows );
    return( null );
}
```

.NET example

The following stored procedure call registers a .NET method called DownloadStats as the script for the download_statistics connection event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_connection_script(  
    'ver1',  
    'download_statistics',  
    'TestScripts.Test.DownloadStats'  
)
```

Following is the C# signature for the call DownloadStats.

```
public void DownloadStats(  
    string user,  
    int warnings,  
    int errors,  
    int deletedRows,  
    int fetchedRows,  
    int downloadRows,  
    int bytes )
```

download_statistics table event

Function Tracks synchronization statistics for download operations by table.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	ml_username	VARCHAR(128). This is the MobiLink user name as specified in your SYNCHRO-NIZATION USER definition.
2	table	VARCHAR(128). The table name.
3	warnings	INTEGER. The number of warnings issued.
4	errors	INTEGER. The number of errors, including handled errors, that occurred.
5	fetches_rows	INTEGER. The number of rows fetched by the download_cursor script.
6	deleted_rows	INTEGER. The number of rows fetched by the download_deletes script.
7	filtered_rows	INTEGER. The number of rows from (6) actually sent to the remote. This reflects download filtering of uploaded values.
8	bytes	INTEGER. The number of bytes sent to the remote as the download.

Default action None.

Description The download_statistics event allows you to gather, for any user and table, statistics on downloads as they apply to that table. The download_statistics table script is called just prior to the commit at the end of the download transaction.

See also [“download_statistics connection event” on page 139](#)

[“upload_statistics connection event” on page 224](#)

[“upload_statistics table event” on page 227](#)

[“synchronization_statistics connection event” on page 202](#)

[“synchronization_statistics table event” on page 205](#)

[“time_statistics connection event” on page 207](#)

[“time_statistics table event” on page 209](#)

[“MobiLink Monitor” \[MobiLink Synchronization User’s Guide, page 297\]](#)

SQL example

The following example comes from an Oracle installation.

```
INSERT INTO download_audit (
    id, user_name, table, warnings, errors,
    deleted_rows, fetched_rows, download_rows, bytes)
VALUES (d_audit.nextval,?,?,?,?,?,?,?,?)
```

Once vital statistics are inserted into the audit table, you may use these statistics to monitor your synchronizations and make optimizations where applicable.

Java example

The following stored procedure call registers a Java method called `downloadStatisticsTable` as the script for the `download_statistics` table event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(
    'ver1',
    'table1',
    'download_statistics',
    'ExamplePackage.ExampleClass.downloadStatisticsTable' )
```

Following is the sample Java method `downloadStatisticsTable`. It prints some statistics for this table to the MobiLink output log.

```
public String downloadStatisticsTable( String user,
    String table, int warnings, int errors, int fetchedRows,
    int deletedRows, int bytes )
{
    java.lang.System.out.println( "download table stats "
        + "table: " + table + "bytes: " + bytes );
    return( null );
}
```

.NET example

The following stored procedure call registers a .NET method called `DownloadTableStats` as the script for the `download_statistics` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script(  
    'ver1',  
    'table1',  
    'download_statistics',  
    'TestScripts.Test.DownloadTableStats'  
)
```

Following is the C# signature for the call DownloadTableStats.

```
public void DownloadTableStats(  
    string user,  
    string table,  
    int warnings,  
    int errors,  
    int deletedRows,  
    int fetchedRows,  
    int downloadRows,  
    int bytes )
```

end_connection connection event

Function	Processes any statements just before the MobiLink synchronization server closes a connection with the consolidated database server, either in preparation to shut down or when a connection is removed from the connection pool. This script is normally used to complete any actions started by the begin_connection script and free any resources acquired by it.
Parameters	None.
Default action	None.
Description	You can use the end_connection script to perform an action of your choice just prior to closing of a connection between the MobiLink synchronization server and the consolidated database server.
See also	“begin_connection connection event” on page 107
SQL example	The following Adaptive Server Anywhere SQL script drops a temporary table that was created by the begin_connection script. Strictly speaking, this table doesn’t need to be dropped explicitly, since ASA will do this automatically when the connection is destroyed. Whether or not a temporary table needs to be dropped explicitly depends on your consolidated database type.

```
call ml_add_connection_script(
  'version 1.0',
  'end_connection',
  'drop table #sync_info' )
```

Java example	The following stored procedure call registers a Java method called endConnection as the script for the end_connection event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.
--------------	---

```
call ml_add_java_connection_script(
  'ver1',
  'end_connection',
  'ExamplePackage.ExampleClass.endConnection' )
```

Following is the sample Java method endConnection. It prints a message to the MobiLink output log. (This might be useful at development time but would slow down a production server.)

```
public String endConnection()
{ java.lang.System.out.println( "ending connection" );
  return( null ); }
```

.NET example

The following stored procedure call registers a .NET method called EndConnection as the script for the end_connection connection event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_connection_script(  
    'ver1',  
    'end_connection',  
    'TestScripts.Test.EndConnection'  
)
```

Following is the C# signature for the call EndConnection.

```
public void EndConnection()
```


end_download connection event

Function	Processes any statements just after the MobiLink synchronization server concludes preparation of the download data.
Parameters	In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [<i>MobiLink Synchronization User’s Guide</i> , page 233] and “SQL-.NET data types” [<i>MobiLink Synchronization User’s Guide</i> , page 261].

Event parameters are optional only if no subsequent parameters are specified. You must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	last_download	TIMESTAMP
2	ml_username	VARCHAR(128)

Default action None.

Description The MobiLink synchronization server executes this script after all rows have been downloaded and, if expecting a download acknowledgement, confirmation of receipt has been received. Download information is processed in a single transaction. The execution of this script is the last non statistical action in this transaction.

The last_download timestamp is the value obtained from the consolidated database during the last successful synchronization immediately prior to the download phase. If the current user has never synchronized successfully, this value is set to 1900-01-01.

See also [“begin_download connection event” on page 110](#)

SQL example The following example shows one possible use of an end_download connection script.

```
DELETE FROM ULEmpCust ec
WHERE ? IS NOT NULL
AND ec.emp_id = ? AND action = 'D'
```

Java example The following stored procedure call registers a Java method called endDownloadConnection as the script for the end_download connection event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_connection_script(
    'ver1',
    'end_download',
    'ExamplePackage.ExampleClass.endDownloadConnection' )
```

Following is the sample Java method `endDownloadConnection`. It uses the current MobiLink connection (saved earlier) to perform an update before the download ends.

```
public String endDownloadConnection(
    Timestamp ts,
    String user )
throws java.sql.SQLException
{ String del_sql =      "DELETE FROM ULEmpCust ec " +
  "WHERE ec.emp_id = '" + user + "' " +
  "AND action = 'D' ";
  execUpdate( _syncConn, del_sql );
  return( null );
}
```

.NET example

The following stored procedure call registers a .NET method called `EndDownload` as the script for the `end_download` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_connection_script(
  'ver1',
  'end_download',
  'TestScripts.Test.EndDownload' )
```

Following is the C# signature for the call `EndDownload`.

```
public void EndDownload(
    DateTime timestamp,
    string user )
```

end_download table event

Function Processes statements related to a specific table just after the MobiLink synchronization server concludes preparing the stream of downloaded inserts, updates, and deletes.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	last_download	TIMESTAMP
2	ml_username	VARCHAR(128)
3	table	VARCHAR(128)

Default action None.

Description The MobiLink synchronization server executes this script after all rows have been downloaded and confirmation of receipt has been received. The download information is prepared in a separate transaction. The execution of this script is the last table-specific, non-statistical action in this transaction.

You can have one end_download script for each table in the remote database.

The last_download timestamp is the value obtained from the consolidated database during the last successful synchronization immediately prior to the download phase. If the current user has never synchronized successfully, this value is set to 1900-01-01.

See also [“begin_download table event” on page 112](#)

[“end_download connection event” on page 147](#)

SQL example The end_download table event is used to perform whatever steps you need performed after downloading a particular table. The following Adaptive Server Anywhere SQL script drops a temporary table created by a prepare_for_download script to hold download rows for the sales_summary table.

```
call ml_add_table_script(  
  'MyCorp 1.0',  
  'sales_summary',  
  'end_download',  
  'drop table #sales_summary_download' )
```

Java example

The following stored procedure call registers a Java method called `endDownloadTable` as the script for the `end_download` table event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script (  
  'ver1',  
  'table1',  
  'end_download',  
  'ExamplePackage.ExampleClass.endDownloadTable' )
```

Following is the sample Java method `endDownloadTable`. It resets the current table member variable.

```
public String endDownloadTable( Timestamp ts,  
  String user, String table )  
{  
  _curTable = null;  
  return( null ); }  
}
```

.NET example

The following stored procedure call registers a .NET method called `EndTableDownload` as the script for the `end_download` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script(  
  'ver1',  
  'table1',  
  'end_download',  
  'TestScripts.Test.EndTableDownload'  
)
```

Following is the C# signature for the call `EndTableDownload`.

```
public void EndTableDownload  
  DateTime timestamp,  
  string user,  
  string table )
```

end_download_deletes table event

Function Processes statements related to a specific table just after preparing a list of rows to be deleted from the specified table in the remote database.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	last_download	TIMESTAMP
2	ml_username	VARCHAR(128)
3	table	VARCHAR(128)

Default action None.

Description This script is executed immediately after preparing a list of rows to be deleted from the named table in the remote database.

You can have one end_download_deletes script for each table in the remote database.

The last_download timestamp is the value obtained from the consolidated database during the last successful synchronization immediately prior to the download phase. If the current user has never synchronized successfully, this value is set to 1900-01-01.

See also [“begin_download_deletes table event” on page 114](#)
[“end_download connection event” on page 147](#)
[“begin_download_rows table event” on page 116](#)
[“end_download_rows table event” on page 153](#)

SQL example You may want to mark a row as deleted on the remote database in this event, using a WHERE clause on the UPDATE that matches the WHERE clause used for your download_delete_cursor.

```

Call ml_add_table_script(
    'version1',
    'Leads',
    'end_download_deletes',
    'UPDATE Leads SET OnRemote = 0
     WHERE LastModified > ?
     AND Owner = ? AND DeleteFlag=1');

```

Java example

The following stored procedure call registers a Java method called `endDownloadDeletes` as the script for the `end_download_deletes` table event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_java_table_script(
    'ver1',
    'table1',
    'end_download_deletes',
    'ExamplePackage.ExampleClass.endDownloadDeletes' )

```

Following is the sample Java method `endDownloadDeletes`. It returns the `end_download_deletes` SQL statement. MobiLink will execute this statement.

```

public String endDownloadDeletes( Timestamp ts,
    String user, String table )
{ return( "UPDATE Leads SET OnRemote = 0
    WHERE LastModified > ?
    AND Owner = ? AND DeleteFlag=1" ); }

```

.NET example

The following stored procedure call registers a .NET method called `EndDownloadDeletes` as the script for the `end_download_deletes` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_dnet_table_script(
    'ver1',
    'table1',
    'end_download_deletes',
    'TestScripts.Test.EndDownloadDeletes'
)

```

Following is the C# signature for the call `EndDownloadDeletes`.

```

public void EndDownloadDeletes(
    DateTime timestamp, string user, string table )

```

end_download_rows table event

Function Processes statements related to a specific table just after preparing a list of rows to be inserted or updated in the specified table in the remote database.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	last_download	TIMESTAMP
2	ml_username	VARCHAR(128)
3	table	VARCHAR(128)

Default action None.

Description This script is executed immediately after preparing the stream of rows to be inserted or updated in the named table in the remote database.

You can have one end_download_rows script for each table in the remote database.

The last_download timestamp is the value obtained from the consolidated database during the last successful synchronization immediately prior to the download phase. If the current user has never synchronized successfully, this value is set to 1900-01-01.

See also [“begin_download_rows table event” on page 116](#)
[“end_download connection event” on page 147](#)
[“end_download_deletes table event” on page 151](#)
[“begin_download_deletes table event” on page 114](#)

SQL example You may want to mark a row as successfully downloaded to the remote database in this event, using a WHERE clause on the UPDATE that matches the WHERE clause used for your download_cursor.

```
call ml_add_table_script(
    'version1',
    'Leads',
    'end_download_rows',
    'UPDATE Leads SET OnRemote = 1 WHERE LastModified > ?
     AND Owner = ? AND DownloadFlag=1');
```

Java example

The following stored procedure call registers a Java method called `endDownloadRows` as the script for the `end_download_rows` table event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(
    'ver1',
    'table1',
    'end_download_rows',
    'ExamplePackage.ExampleClass.endDownloadRows' )
```

Following is the sample Java method `endDownloadRows`. It prints a message to the MobiLink output log. (This might be useful at development time but would slow down a production server.)

```
public String endDownloadRows(
    Timestamp ts,
    String user,
    String table )
{
    java.lang.System.out.println( "Done downloading
    inserts and updates for table " + table );
    return( null ); }
}
```

.NET example

The following stored procedure call registers a .NET method called `EndDownloadRows` as the script for the `end_download_rows` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script(
    'ver1', 'table1', 'end_download_rows',
    'TestScripts.Test.EndDownloadRows'
)
```

Following is the C# signature for the call `EndDownloadRows`.

```
public void EndDownloadRows(
    DateTime timestamp,
    string user,
    string table )
```


end_publication connection event

Function Provides useful information about the publication(s) being synchronized.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Item	Parameter	Description
1	generation_number	INTEGER. If your deployment does not use file-based downloads, this parameter can be ignored. The default is 1.
2	ml_username	VARCHAR(128). If an UltraLite remote is synchronizing with UL_SYNC_ALL, this event is invoked once with the name ‘unknown’.
3	publication_name	VARCHAR(128)
4	last_upload	TIMESTAMP. Last successful upload.
5	last_download	TIMESTAMP. Last successful download.

Default action None.

Description This event lets you design synchronization logic based on the publications currently being synchronized. This event is invoked in the same transaction as the end_synchronization event, and is invoked before the end_synchronization event. It is invoked once per publication being synchronized.

If the current synchronization successfully applied an upload, the last_upload parameter will contain the time this latest upload was applied. If the current synchronization has a successful download acknowledgement, the last_download time will contain the time this latest download was generated. This is the same value that was passed to the download scripts as the last download timestamp.

Generation number The generation_number parameter is specifically for file-based downloads. The output value of the generation number is passed from the begin_publication script to the end_publication script. The meaning of the

generation_number depends on whether the current synchronization is being used to create a download file, or whether the current synchronization has an upload.

In file-based downloads, generation numbers are used to force an upload before the download. The number is stored in the download file.

See also

[“begin_publication connection event” on page 118](#)

“File-Based Downloads” [*MobiLink Synchronization User’s Guide*, page 117]

SQL example

You may want to record the information for each publication being synchronized:

```
call ml_add_connection_script(  
    'version1',  
    'end_publication',  
    '{call RecordPubEndSync( ?, ?, ?, ?, ? )}' );
```

Java example

The following stored procedure call registers a Java method called endPublication as the script for the begin_publication connection event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_connection_script(  
    'ver1',  
    'end_publication',  
    'ExamplePackage.ExampleClass.endPublication' )
```

Following is the sample Java method endPublication. It outputs a message to the MobiLink log.

```
public String endPublication(  
    ianywhere.ml.script.InOutInteger generation_number,  
    String user,  
    String pub_name,  
    Timestamp last_upload,  
    Timestamp last_download )  
{ System.out.println(  
    "Finished synchronizing publication " + pub_name );  
    return( null ); }
```

.NET example

The following stored procedure call registers a .NET method called EndPub as the script for the end_publication connection event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_connection_script( 'ver1',  
    'end_publication',  
    'TestScripts.Test.EndPub'  
)
```

Following is the C# signature for EndPub.

```
public void EndPub(  
    ref int generation_number,  
    string user,  
    string pub_name,  
    DateTime last_upload,  
    DateTime last_download )  
{  
    Console.Write(  
        "Finished synchronizing publication " + pub_name );  
}
```

end_synchronization connection event

Function Processes any statements at the time an application disconnects from the MobiLink synchronization server upon completion of the synchronization process.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Item	Parameter	Description
1	ml_username	VARCHAR(128)
2	sync_ok	INTEGER. This value is 1 for a successful synchronization and 0 for an unsuccessful synchronization.

Default action None.

Description The MobiLink synchronization server executes this script after synchronization is complete and, if expecting a download acknowledgement, the MobiLink client has returned confirmation of receipt of the download stream.

This script is executed within a separate transaction after the download transaction. It is useful for maintaining statistics.

See also [“begin_synchronization connection event” on page 121](#)
[“begin_synchronization table event” on page 123](#)
[“end_synchronization table event” on page 160](#)

SQL example The following Adaptive Server Anywhere SQL script calls a stored procedure that records the end time of the synchronization attempt along with its success or failure status:

```
call ml_add_connection_script(  
    'ver1',  
    'end_synchronization',  
    'call RecordEndOfSyncAttempt(?,?)' )
```

Java example The following stored procedure call registers a Java method called endSynchronizationConnection as the script for the end_synchronization event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_connection_script(  
    'ver1',  
    'end_synchronization',  
    'ExamplePackage.ExampleClass.endSynchronizationConnection'  
)
```

Following is the content of the sample Java method `endSynchronizationConnection`. It uses the JDBC connection to execute an update.

```
public String endSynchronizationConnection(  
    String user )  
    throws java.sql.SQLException  
{    execUpdate( _syncConn, "UPDATE sync_count set cnt =  
        count + 1 where user_id = '" + user + "' " );  
    return( null ); }
```

.NET example

The following stored procedure call registers a .NET method called `EndSync` as the script for the `end_synchronization` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_connection_script(  
    'ver1',  
    'end_synchronization',  
    'TestScripts.Test.EndSync'  
)
```

Following is the C# signature for the call `EndSync`.

```
public void EndSync( string user )
```

end_synchronization table event

Function Processes statements related to a specific table at the time an application disconnects from the MobiLink synchronization server upon completion of the synchronization process.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	ml_username	VARCHAR(128)
2	table	VARCHAR(128)
3	sync_ok	INTEGER. This value is 1 for a successful synchronization and 0 for an unsuccessful synchronization.

Default action None.

Description The MobiLink synchronization server executes this script after an application has synchronized and is about to disconnect from the MobiLink synchronization server, and before the connection level script of the same name.

You can have one end_synchronization script for each table in the remote database.

See also [“begin_synchronization table event” on page 123](#)
[“end_synchronization connection event” on page 158](#)
[“end_synchronization table event” on page 160](#)

SQL example The following Adaptive Server Anywhere SQL script drops a temporary table created by the begin_synchronization script:

```
call ml_add_table_script(  
  'ver1',  
  'sales_order',  
  'end_synchronization',  
  'drop table #sales_order' )
```

Java example

The following stored procedure call registers a Java method called `endSynchronizationTable` as the script for the `end_synchronization` table event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(  
    'ver1',  
    'table1',  
    'end_synchronization',  
    'ExamplePackage.ExampleClass.endSynchronizationTable' )
```

Following is the sample Java method `endSynchronizationTable`. It takes no action. MobiLink interprets `NULL` as no script.

```
public String endSynchronizationTable( String user,  
String table )  
{ return( null ); }
```

.NET example

The following stored procedure call registers a .NET method called `EndTableSync` as the script for the `end_synchronization` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script(  
    'ver1', 'table1', 'end_synchronization',  
    'TestScripts.Test.EndTableSync'  
)
```

Following is the C# signature for the call `EndTableSync`.

```
public void EndTableSync( string user, string table )
```

end_upload connection event

Function Processes any statements just after the MobiLink synchronization server concludes processing uploaded inserts, updates, and deletes.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [MobiLink Synchronization User’s Guide, page 233] and “SQL-.NET data types” [MobiLink Synchronization User’s Guide, page 261].

Item	Parameter	Description
1	ml_username	VARCHAR(128)

Default action None.

Description The MobiLink synchronization server executes this script as the last step in the processing of uploaded information. Upload information is processed in a single transaction. The execution of this script is the last action in this transaction before statistical scripts.

See also [“begin_upload connection event” on page 125](#)
[“end_upload table event” on page 164](#)

SQL example The following statements define a stored procedure and an end_upload script suited to the CustDB sample application from an Oracle installation.

```
CREATE OR REPLACE PROCEDURE ULCustomerIDPool_maintain(  
    SyncUserID IN integer )  
AS  
    pool_count INTEGER;  
    pool_max   INTEGER;  
BEGIN  
    -- Determine how many ids to add to the pool  
  
    SELECT COUNT(*)  
        INTO pool_count  
        FROM ULCustomerIDPool  
        WHERE pool_emp_id = SyncUserID;  
    -- Determine the current Customer id max  
  
    SELECT MAX(pool_cust_id)  
        INTO pool_max  
        FROM ULCustomerIDPool;  
    -- Top up the pool with new ids
```



```

WHILE pool_count < 20 LOOP
    pool_max := pool_max + 1;
    INSERT INTO ULCustomerIDPool(
        pool_cust_id, pool_emp_id )
        VALUES ( pool_max, SyncUserID );
    pool_count := pool_count + 1;
END LOOP;
END;
ml_add_table_script(
    'custdb',
    'ULCustomerIDPool',
    'end_upload', -
    ULCustomerIDPool_maintain( ? );)

```

Java example

The following stored procedure call registers a Java method called `endUploadConnection` as the script for the `end_upload` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_java_connection_script(
    'ver1',
    'end_upload',
    'ExamplePackage.ExampleClass.endUploadConnection' )

```

Following is the sample Java method `endUploadConnection`. It calls a method to perform operations on the database.

```

public String endUploadConnection( String user )
{ // clean up new and old tables
    Iterator two_iter = _tables_with_ops.iterator();
    while( two_iter.hasNext() )
    { TableInfo cur_table = (TableInfo)two_iter.next();
      dumpTableOps( _sync_conn, cur_table ); }
    _tables_with_ops.clear(); }

```

.NET example

The following stored procedure call registers a .NET method called `EndUpload` as the script for the `end_upload` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_dnet_connection_script(
    'ver1',
    'end_upload',
    'TestScripts.Test.EndUpload'
)

```

Following is the C# signature for the call `EndUpload`.

```

public void EndUpload( string user )

```

end_upload table event

Function Processes statements related to a specific table just after the MobiLink synchronization server concludes processing the stream of uploaded inserts, updates, and deletions.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. You must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	ml_username	VARCHAR(128)
2	table	VARCHAR(128)

Default action None.

Description The MobiLink synchronization server executes this script as the last step in the processing of uploaded information. Upload information is processed in a separate transaction. The execution of this script is the last table-specific action in this transaction.

You can have one end_upload script for each table in the remote database.

See also [“begin_upload table event” on page 127](#)
[“end_upload connection event” on page 162](#)

SQL example The event can be used to clean up anything that may still exist in the database after the upload processing is done for a particular table.

```
Call ml_add_table_script(  
    'version1',  
    'Leads',  
    'end_upload',  
    'DELETE FROM T_Leads');
```

Java example The following stored procedure call registers a Java method called endUploadTable as the script for the end_upload table event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(
    'ver1',
    'table1'
    'end_upload',
    'ExamplePackage.ExampleClass.endUploadTable' )
```

Following is the sample Java method `endUploadTable`. It generates a delete for a table with a name related to the passing-in table name.

```
public String endUploadTable(    String user,
String table)
{    return( "DELETE from '" + table + "_temp'" ); }
```

.NET example

The following stored procedure call registers a .NET method called `EndTableUpload` as the script for the `end_upload` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script(
    'ver1',
    'table1',
    'end_upload',
    'TestScripts.Test.EndTableUpload'
)
```

Following is the C# signature for the call `EndTableUpload`.

```
public void EndTableUpload(
    string user, string table )
```

The following C# example moves rows inserted into a temporary table into the table passed into the script.

```
public void EndUpload( string user, string table )
{
    DBCommand stmt = curConn.CreateCommand();

    // move the uploaded rows to the destination table
    stmt.CommandText = "INSERT INTO "
        + table
        + " SELECT * FROM dnet_ul_temp";
    stmt.ExecuteNonQuery();
    stmt.Close();
}
```

end_upload_deletes table event

Function Processes statements related to a specific table just after applying deletes uploaded from the specified table in the remote database.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. You must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	ml_username	VARCHAR(128)
2	table	VARCHAR(128)

Default action None.

Description This script is run immediately after applying the changes that result from rows deleted in the remote table named in the second parameter.

You can have one end_upload_deletes script for each table in the remote database.

See also [“begin_upload_deletes table event” on page 129](#)

SQL example You can use this event to process rows deleted during the upload stream on an intermediate table. You can compare the rows in the base table with rows in the intermediate table and decide what to do with the deleted row.

```
Call ml_add_table_script(
    'version1',
    'Leads',
    'end_uploads_deletes',
    'call EndUploadDeletesLeads()');
Create procedure EndUploadDeletesLeads ( )
Begin
    FOR names AS curs CURSOR FOR
        SELECT LeadID
        FROM Leads
        WHERE LeadID NOT IN (SELECT LeadID FROM T_Leads);
    DO
        CALL decide_what_to_do( LeadID );
    END FOR;
end
```

Java example The following stored procedure call registers a Java method called endUploadDeletes as the script for the end_upload_deletes table event when

synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(  
    'ver1',  
    'table1',  
    'end_upload_deletes',  
    'ExamplePackage.ExampleClass.endUploadDeletes' )
```

Following is the sample Java method `endUploadDeletes`. It calls a Java method that manipulates the database.

```
public String endUploadDeletes( String user,  
    String table )  
    throws java.sql.SQLException  
{ processUploadedDeletes( _syncConn, table );  
    return( null ); }
```

.NET example

The following stored procedure call registers a .NET method called `EndUploadDeletes` as the script for the `end_upload_deletes` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script(  
    'ver1',  
    'table1',  
    'end_upload_deletes',  
    'TestScripts.Test.EndUploadDeletes'  
)
```

Following is the C# signature for the call `EndUploadDeletes`.

```
public void EndUploadDeletes( string user, string table )
```

end_upload_rows table event

Function Processes statements related to a specific table just after applying uploaded inserts and updates from the specified table in the remote database.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. You must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	ml_username	VARCHAR(128)
2	table	VARCHAR(128)

Default action None.

Description Uploaded information can require inserting or updating rows in the consolidated database. This script is run immediately after applying the changes that result from modifications to the remote table named in the second parameter.

You can have one end_upload_rows script for each table in the remote database.

See also [“begin_upload_rows table event” on page 131](#)

SQL example You use this event to process rows deleted during the upload stream on an intermediate table. You can compare the rows in the base table with the rows in the intermediate table and decide what to do with the deleted row.

```
Call ml_add_table_script(  
    'version1',  
    'Leads',  
    'end_uploads_deletes',  
    'call EndUploadDeletesLeads()');  
Create procedure EndUploadDeletesLeads ( )  
Begin  
    FOR names AS curs CURSOR FOR  
        SELECT LeadID FROM Leads  
            WHERE LeadID NOT IN (select LeadID from T_Leads);  
    DO  
        CALL decide_what_to_do( LeadID );  
    END FOR;  
end
```

Java example

The following stored procedure call registers a Java method called `endUploadRows` as the script for the `end_upload_rows` table event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(  
    'ver1',  
    'table1',  
    'end_upload_rows',  
    'ExamplePackage.ExampleClass.endUploadRows' )
```

Following is the sample Java method `endUploadRows`. It calls a Java method that manipulates the database.

```
public String endUploadRows(    String user,  
String table )  
    throws java.sql.SQLException  
{    processUploadedRows( _syncConn, table );  
    return( null ); }
```

.NET example

The following stored procedure call registers a .NET method called `EndUploadRows` as the script for the `end_upload_rows` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script(  
    'ver1',  
    'table1',  
    'end_upload_rows',  
    'TestScripts.Test.EndUploadRows'  
)
```

Following is the C# signature for the call `EndUploadRows`.

```
public void EndUploadRows(  
    string user,  
    string table )
```

example_upload_cursor table event

Function Provides an event that the MobiLink synchronization server does not use during processing of the upload stream to handle rows inserted into the remote database. The event is not called.

Parameters

Item	Parameter
1	column 1
2	column 2
...	...

Description The statement based example_upload_cursor script performs direct inserts of column values identical to those specified in the example_upload_cursor statement. The example_upload_cursor event is not called by MobiLink.

SQL example The script is not called. If it were called, it would insert the values into a table named Customer in the consolidated database. The final column of the table identifies the Customer as active. The final column does not appear in the remote database.

```
SELECT cust_id, name, rep_id
FROM customer
WHERE cust_id=?
```


example_upload_delete table event

Function Processes the upload stream to handle rows deleted from the remote database. The script is not called by MobiLink.

Parameters

Item	Parameter
1	column 1
2	column 2
...	...

Description The statement based example_upload_delete script handles rows that are deleted in the remote database. The action taken at the consolidated database can be a DELETE statement, but need not be.

See also [“upload_delete table event” on page 214](#)

SQL example This example marks customers that are deleted from the remote database as inactive.

```
UPDATE Customer
SET active = 0
WHERE cust_id=?
```

example_upload_insert table event

Function	Provides an event that the MobiLink synchronization server uses during processing of the upload stream to handle rows inserted into the remote database.								
Parameters	<table><tr><th>Item</th><th>Parameter</th></tr><tr><td>1</td><td>column 1</td></tr><tr><td>2</td><td>column 2</td></tr><tr><td>...</td><td>...</td></tr></table>	Item	Parameter	1	column 1	2	column 2
Item	Parameter								
1	column 1								
2	column 2								
...	...								
Description	<p>The statement based example_upload_insert script performs direct inserts of column values identical to those specified in the upload_insert statement.</p> <p>The example_upload_insert event is not called.</p>								
See also	“upload_insert table event” on page 218								
SQL example	<p>The script is not called. But if called, it would insert the values into a table named Customer in the consolidated database. The final column of the table identifies the Customer as active. The final column does not appear in the remote database.</p>								

```
INSERT INTO Customer( cust_id, name, rep_id )
VALUES ( ?, ?, ? )
```

example_upload_update table event

Function An example event for the upload stream to handle rows updated at the remote database. The example script is not called by MobiLink but is identical in form to the upload_update event.

Parameters

Clause	Parameters
SET	column 1 column 2 ...
WHERE	primary key 1 primary key 2 ...

Description You create an example_upload_update event script by using the option -za in the dbmlsync command line.

See also [“upload_update table event” on page 231](#)

SQL example This example handles updates made to the Customer table in the remote database. The script updates the values in a table named Customer in the consolidated database. Note: The script is never called and is only an example script.

```
UPDATE Customer
SET name=?, rep_id=?
WHERE cust_id=?
```

handle_error connection event

Function Executed whenever the MobiLink synchronization server encounters a SQL error.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	action_code	INTEGER. This is an IN-OUT parameter.
2	error_code	INTEGER
3	error_message	TEXT
4	ml_username	VARCHAR(128)
5	table	VARCHAR(128). If the script is not a table script, the table name is NULL.

Default action When no handle_error script is defined or this script causes an error, the default action code is 3000: rollback the current transaction and cancel the current synchronization.

Description The MobiLink synchronization server sends in the current action_code. Initially, this is set to 3000 for each set of errors caused by a single SQL operation. Usually, there is only one error per SQL operation, but there may be more. This handle_error script is called once per error in the set. The action code passed into the first error is 3000. Subsequent calls are passed in the action code returned by the previous call. MobiLink will use the numerically highest value returned from multiple calls.

You can modify the action code in the script, and return a value instructing MobiLink how to proceed. The action code parameter takes one of the following values:

- ◆ **1000** Skip the current row and continue processing.
- ◆ **3000** Rollback the current transaction and cancel the current

synchronization. This is the default action code, and is used when no `handle_error` script is defined or this script causes an error.

- ◆ **4000** Rollback the current transaction, cancel the synchronization, and shut down the MobiLink synchronization server.

SQL scripts for the `handle_error` event must be implemented as stored procedures.

The MobiLink synchronization server executes this script whenever it encounters an error during the synchronization process. The error codes and message allow you to identify the nature of the error. If the error happened as part of synchronization, the user name is supplied. Otherwise, this value is NULL.

If the error happened while manipulating a particular table, the table name is supplied. Otherwise, this value is NULL. The table name is the name of a table in the client application. This name may or may not have a direct counterpart in the consolidated database, depending upon the design of the synchronization system.

The action code tells the MobiLink synchronization server what to do next. Before it calls this script, the MobiLink synchronization server sets the action code to a default value, which depends upon the severity of the error. Your script may modify this value. Your script must return or set an action code.

You can return a value from the `handle_error` script in two ways.

- ◆ Pass the action parameter to an OUTPUT parameter of a procedure:

```
CALL my_handle_error( ?, ?, ?, ?, ? )
```

- ◆ Set the action code via a procedure or function return value:

```
? = CALL my_handle_error( ?, ?, ?, ? )
```

Most DBMSs use the RETURN statement to set the return value from a procedure or function.

The CustDB sample application contains error handlers for various database-management systems.

See also

[“report_error connection event” on page 194](#)

[“report_odbc_error connection event” on page 196](#)

[“handle_odbc_error connection event” on page 177](#)

SQL example

The following example works with an Adaptive Server Anywhere consolidated database. It allows your application to ignore redundant inserts.

```

CREATE PROCEDURE ULHandleError(
    INOUT action integer,
    IN error_code integer,
    IN error_message varchar(1000),
    IN user_name varchar(128),
    IN table_name varchar(128) )
BEGIN
    -- -196 is SQLE_INDEX_NOT_UNIQUE
    -- -194 is SQLE_INVALID_FOREIGN_KEY
    if error_code = -196 or error_code = -194 then
        -- ignore the error and keep going
        SET action = 1000;
    else
        -- abort the synchronization
        SET action = 3000;
    end if;
END

```

Java example

The following stored procedure call registers a Java method called `handleError` as the script for the `handle_error` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_java_connection_script(
    'ver1',
    'handle_error',
    'ExamplePackage.ExampleClass.handleError' )

```

Following is the sample Java method `handleError`. It processes an error based on the data that is passed in. It also determines the resulting error code.

```

public String handleError(
    anywhere.ml.script.InOutInteger actionCode,
    int errorCode,
    String errorMessage,
    String user,
    String table )
{
    int new_ac;
    if( user == null )
    {
        new_ac = handleNonSyncError( errorCode,
            errorMessage );
    }
    else if( table == null )
    {
        new_ac = handleConnectionError( errorCode,
            errorMessage, user );
    }
    else
    {
        new_ac = handleTableError( errorCode,
            errorMessage, user, table );
    }
    // keep the most serious action code
    if( actionCode.getValue() < new_ac )
    {
        actionCode.setValue( new_ac );
    }
    return( null );
}

```

handle_odbc_error connection event

Function Executed whenever the MobiLink synchronization server encounters an error triggered by the ODBC Driver Manager.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	action_code	INTEGER. This is an INOUT parameter.
2	ODBC_state	VARCHAR(5)
3	error_message	TEXT
4	ml_username	VARCHAR(128)
5	table	VARCHAR(128)

Default action The MobiLink synchronization server selects a default action code. You can modify the action code in the script, and return a value instructing MobiLink how to proceed. The action code parameter takes one of the following values:

- ◆ **1000** Skip the current row and continue processing.
- ◆ **3000** Rollback the current transaction and cancel the current synchronization. This is the default action code, and is used when no handle_error script is defined or this script causes an error.
- ◆ **4000** Rollback the current transaction, cancel the synchronization, and shut down the MobiLink synchronization server.

Description The MobiLink synchronization server executes this script whenever it encounters an error flagged by the ODBC Driver Manager during the synchronization process. The error codes allow you to identify the nature of the error.

The action code tells the MobiLink synchronization server what to do next. Before it calls this script, the MobiLink synchronization server sets the action code to a default value, which depends upon the severity of the error.

Your script may modify this value. Your script must return or set an action code.

The `handle_odbc_error` script is called after the `handle_error` and `report_error` scripts, and before the `report_odbc_error` script.

When only one, but not both, error-handling script is defined, the return value from that script decides error behavior. When both error-handling scripts are defined, the MobiLink synchronization server uses the numerically highest action code. If both `handle_error` and `handle_ODBC_error` are defined, MobiLink uses the numerically highest action code returned from all calls.

See also

[“handle_error connection event” on page 174](#)

[“report_error connection event” on page 194](#)

[“report_odbc_error connection event” on page 196](#)

SQL example

The following example works with an Adaptive Server Anywhere consolidated database. It allows your application to ignore ODBC integrity constraint violations.

```
call ml_add_connection_script(
  'ver1',
  'handle_odbc_error',
  'call HandleODBCError( ?, ?, ?, ?, ? )' )
CREATE PROCEDURE HandleODBCError( INOUT action integer,
IN odbc_state varchar(5), IN error_message varchar(1000),
IN user_name varchar(128), IN table_name varchar(128) )
BEGIN
  if odbc_state = '23000' then
    -- ignore the error and keep going
    SET action = 1000;
  else
    -- abort the synchronization
    SET action = 3000;
  end if;
END
```

Java example

The following stored procedure call registers a Java method called `handleODBCError` as the script for the `handle_odbc_error` event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_connection_script( 'ver1', 'handle_odbc_error',
  'ExamplePackage.ExampleClass.handleODBCError' )
```

Following is the sample Java method `handleODBCError`. It processes an error based on the data that is passed in. It also determines the resulting error code.


```
public String handleODBCError(
    ianywhere.ml.script.InOutInteger actionCode,
    String ODBCState,
    String errorMessage,
    String user,
    String table )
{   int new_ac;
    if( user == null )
    {   new_ac = handleNonSyncError( ODBCState,
        errorMessage ); }

    else if( table == null )
    {   new_ac = handleConnectionError( ODBCState,
        errorMessage, user ); }
    else {   new_ac = handleTableError( ODBCState,
        errorMessage, user, table ); }
    // keep the most serious action code
    if( actionCode.getValue() < new_ac )
    {   actionCode.setValue( new_ac ); }
    return( null ); }
```

modify_last_download_timestamp connection event

Function The script can be used to modify the last_download timestamp for the current synchronization.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. You must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	last_download_timestamp	TIMESTAMP. This is an INOUT parameter.
2	ml_username	VARCHAR(128)

Default action None.

Description Use this script when you want to modify the last download timestamp for the current synchronization. If this script is defined, the MobiLink synchronization server calls this script and uses the modified last_download timestamp as the last_download timestamp passed to the download scripts.

SQL scripts for the modify_last_download_timestamp event must be implemented as stored procedures. The MobiLink synchronization server passes in the last_download_timestamp as the first parameter to the stored procedure, and replaces the timestamp by the first value passed out by the stored procedure.

For example, if you defined the following download_cursor script, the MobiLink synchronization server would replace the ? with the value of the modified last download timestamp before executing the SELECT statement:

```
SELECT pk, c2, c3 FROM test WHERE last_modified > ?
```

This script is executed just before the prepare_for_download script, in the same transaction.

SQL example The following example downloads everything from one day ago, regardless of whether the databases were synchronized since then.

First, create a procedure for your Adaptive Server Anywhere consolidated database:

```

CREATE PROCEDURE ModifyLastDownloadTimestamp(
    inout last_download_time TIMESTAMP,
    in user_name VARCHAR(128) )
BEGIN
    SELECT dateadd(day, -1, last_download_time )
    INTO last_download_time
END

```

Second, install the script into your Adaptive Server Anywhere consolidated database:

```

call ml_add_connection_script(
    'modify_ts_test',
    'modify_last_download_timestamp',
    'call ModifyLastDownloadTimestamp ( ?, ? )' )

```

Java example

The following stored procedure call registers a Java method called `modifyLastDownloadTimestamp` as the script for the `modify_last_download_timestamp` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_java_connection_script(
    'ver1',
    'modify_last_download_timestamp',
    'ExamplePackage.ExampleClass.modifyLastDownloadTimestamp' )

```

Following is the sample Java method `modifyLastDownloadTimestamp`. It prints the current and new timestamp and modifies the timestamp that is passed in.

```

public String modifyLastDownloadTimestamp(
    Timestamp last_download_time,
    String user_name )
{
    java.lang.System.out.println( "old date: " +
        last_download_time.toString() );
    last_download_time.setDate(
        last_download_time.getDate() -1 );
    java.lang.System.out.println( "new date: " +
        last_download_time.toString() );
    return( null ); }

```

modify_next_last_download_timestamp connection event

Function The script can be used to modify the last_download timestamp for the next synchronization.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	download_timestamp	TIMESTAMP. This is an INOUT parameter.
2	last_download_timestamp	TIMESTAMP
3	ml_username	VARCHAR(128)

Default action None.

Description Use this script when you want to modify the last download timestamp for the next synchronization. If this script is defined, the MobiLink synchronization server calls this script and sends the next last download timestamp down to the remote, which will send it as part of the next synchronization.

SQL scripts for the modify_next_last_download_timestamp event must be implemented as stored procedures. The MobiLink synchronization server passes in the download_timestamp as the first parameter to the stored procedure, and replaces the timestamp by the first value passed out by the stored procedure.

You can use this script to modify the download timestamp that the MobiLink synchronization server sends to the MobiLink client. If the client is dbmlsync, the timestamp is stored in the SYSSYNC system table.

This script is executed in the download transaction, after downloading user tables.

SQL example The following example shows one application of this script. First, create a procedure for your Adaptive Server Anywhere consolidated database:

```

CREATE PROCEDURE ModifyNextDownloadTimestamp(
    inout download_timestamp TIMESTAMP ,
    in last_download TIMESTAMP ,
    in user_name VARCHAR(128) )
BEGIN
    SELECT dateadd(hour, -1, download_timestamp )
    INTO download_timestamp
END

```

Second, install the script into your Adaptive Server Anywhere consolidated database:

```

call ml_add_connection_script(
    'modify_ts_test',
    'modify_next_last_download_timestamp',
    'call ModifyNextDownloadTimestamp ( ?, ?, ? )' )

```

Java example

The following stored procedure call registers a Java method called `modifyNextDownloadTimestamp` as the script for the `modify_next_last_download_timestamp` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```

call ml_add_java_connection_script(
    'ver1',
    'modify_next_last_download_timestamp',
    'ExamplePackage.ExampleClass.modifyNextDownloadTimestamp' )

```

Following is the sample Java method `modifyNextDownloadTimestamp`. It sets the download timestamp back an hour.

```

public String modifyNextDownloadTimestamp(
    Timestamp download_timestamp,
    Timestamp last_download,
    String user_name )
{
    download_timestamp.setHours(
        download_timestamp.getHours() -1 );
    return( null ); }

```

modify_user connection event

Function Provide the MobiLink user name.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Item	Parameter	Description
1	ml_username	VARCHAR(128). This is an INOUT parameter.

Default action None.

Description The MobiLink server provides the user name as a parameter when it calls scripts; the user name is sent by the MobiLink client. In some cases, you may want to have an alternate user name. This script allows you to modify the user name used in calling MobiLink scripts.

The ml_username parameter must be long enough to hold the user name.

SQL scripts for the modify_last_download_timestamp event must be implemented as stored procedures.

See also [“authenticate_user connection event” on page 100](#)
[“authenticate_user_hashed connection event” on page 104](#)

SQL example The following example works with an Adaptive Server Anywhere consolidated database. It maps a remote database user name to the id of the user using the device, by using a mapping table called user_device. This technique can be used when the same person has multiple remotes (such as a PDA and a laptop) requiring the same synchronization logic (based on the user’s name or id).

```
call ml_add_connection_script(
    'ver1',
    'modify_user',
    'call ModifyUser( ? )' )
CREATE PROCEDURE ModifyUser( INOUT u_name varchar(128) )
BEGIN
    select user_name
    into u_name
    from user_device
    where device_name = u_name
END
```

Java example The following stored procedure call registers a Java method called

modifyUser as the script for the modify_user connection event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_connection_script(
    'ver1',
    'modify_user',
    'ExamplePackage.ExampleClass.modifyUser' )
```

Following is the sample Java method modifyUser. It gets the user ID from the database and then uses it to set the user name.

```
public void ModifyUser( InOutString io_user_name )
    throws SQLException
{
    Statement uid_select = curConn.createStatement();
    ResultSet uid_result = uid_select.executeQuery(
        "select rep_id from SalesRep where name = '" +
        io_user_name.getValue() + "' " );
    uid_result.next();
    io_user_name.setValue(
        java.lang.Integer.toString(uid_result.getInt( 1 ))
    );
    uid_result.close();
    uid_select.close();
    return; }
}
```

.NET example

The following stored procedure call registers a .NET method called ModUser as the script for the modify_user connection event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_connection_script(
    'ver1',
    'modify_user',
    'TestScripts.Test.ModUser'
)
```

Following is the C# signature for the call ModUser.

```
public void ModUser( string user )
```

new_row_cursor cursor event (deprecated)

Function Defines the insert cursor that the MobiLink synchronization server uses to insert the new values of rows that were updated in the remote database, but conflict with values presently in the consolidated database.

Use statement-based events for uploads

This script has been deprecated. Use the statement-based event upload_new_row_insert instead. Support for the new_row_cursor event is likely to be removed from future releases.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Item	Parameter	Description
1	ml_username	VARCHAR(128)

Default action None.

Description When a row is updated on a remote database, the MobiLink client saves a copy of the original values. The client sends both old and new values to the MobiLink synchronization server. Also used to input an INSERT operation in forced conflict mode.

When the MobiLink synchronization server receives an updated row, it compares the original values with the present values in the consolidated database, using the upload_cursor. If the old uploaded values do not match the current value in the consolidated database, the row conflicts. Instead of updating the row, the MobiLink synchronization server inserts both old and new values into the consolidated database using the old_row_cursor and the new_row_cursor, respectively.

The MobiLink synchronization server uses a cursor to insert the new uploaded values from conflicting rows into the consolidated database. This script contains the SELECT statement used to define this cursor.

It is common practice to use temporary tables to store the old and new versions of conflicting rows. You can create these temporary tables in an earlier script.

You can have one new_row_cursor script for each table in the remote database.

Normally, the columns in the select list must match those in the client table

in both order and type. However, the MobiLink synchronization server permits you to add one extra column. If you do so, the MobiLink synchronization server automatically inserts the user name into the first column, then proceeds to insert the new row values using the remaining columns, as usual.

Note

The script is ignored if any of the following scripts are defined for the same table: `upload_insert`, `upload_update`, `upload_delete`, `upload_fetch`, `upload_new_row_insert`, `upload_old_row_insert`.

See also

[“upload_new_row_insert table event” on page 220](#)

[“Handling conflicts” \[MobiLink Synchronization User’s Guide, page 90\]](#)

[“resolve_conflict table event” on page 199](#)

SQL example

The following `SELECT` statement defines a `new_row_cursor` script suited to the CustDB sample application.

```
SELECT order_id, cust_id, prod_id, emp_id,
       disc, quant, notes, status
FROM ULNewOrder FOR update
```

The primary key of the `ULOrder` table is `order_id`.

The following `SELECT` statement could instead be used for the same client table. This variation includes the permitted one extra row. The MobiLink synchronization server automatically stores the user name in the first column.

```
SELECT user_name, order_id, cust_id, prod_id,
       emp_id, disc, quant, notes, status
FROM ULNewOrder FOR update
```

Java example

This script must return valid SQL.

The following stored procedure call registers a Java method called `newRowCursor` as the script for the `new_row_cursor` event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(
    'ver1',
    'table1',
    'new_row_cursor',
    'ExamplePackage.ExampleClass.newRowCursor' )
```

Following is the sample Java method `newRowCursor`. It dynamically generates a new row cursor statement by calling a Java method.

```
public String newRowCursor()  
{ return( getRowCursor ( _curTable ) ); }
```

old_row_cursor cursor event (deprecated)

Function Defines the cursor that the MobiLink synchronization server uses to insert the old values of rows that were updated in the remote database, but that conflict with values presently in the consolidated database. The event is also used to insert the values of deleted rows when in forced conflict mode.

Use statement-based events for uploads

This script has been deprecated. Use the statement-based event `upload_old_row_insert` instead. Support for the `old_row_cursor` event is likely to be removed from future releases.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Item	Parameter	Description
1	ml_username	VARCHAR(128)

Default action None.

Description When a row is updated on a remote database, the MobiLink client saves a copy of the original values. The client sends both old and new values to the MobiLink synchronization server.

When the MobiLink synchronization server receives an updated row, it compares the original values with the present values in the consolidated database, using the `upload_cursor` cursor event. If the old uploaded values do not match the current values in the consolidated database, the row conflicts. Instead of updating the row, the MobiLink synchronization server inserts both old and new values into the consolidated database using the `old_row_cursor` event and the `new_row_cursor` event.

It is common practice to use temporary tables to store the old and new versions of conflicting rows. In Adaptive Server Anywhere, you can create these tables in an earlier script. Some non-ASA consolidated databases support temporary tables, but they usually differ significantly from the temporary tables offered by ASA. Consult your DBMS documentation for details. An alternative to a temporary table is a base table with an extra column for the MobiLink user name. This effectively partitions the rows of the base table between concurrent synchronizations.

The MobiLink synchronization server uses a cursor to insert the old uploaded values from conflicting rows into the consolidated database. This

script contains the SELECT statement used to define this cursor.

You can have one `old_row_cursor` script for each table in the remote database.

Normally, the columns in the SELECT list must match those in the client table in both order and type. However, the MobiLink synchronization server permits you to add one extra column. If you do so, the MobiLink synchronization server automatically inserts the user name into the first column, then proceeds to insert the old row values using the remaining columns, as usual.

See also

[“upload_old_row_insert table event” on page 222](#)

[“Handling conflicts”](#) [*MobiLink Synchronization User’s Guide*, page 90]

[“resolve_conflict table event” on page 199](#)

SQL example

The following SELECT statement defines an `old_row_cursor` script suited to the CustDB sample application for an Oracle installation. The primary key of the ULOrder table is `order_id`.

```
SELECT order_id, cust_id, prod_id, emp_id,  
       disc, quant, notes, status  
FROM ULOldOrder
```

The following SELECT statement could instead be used for the same client table. This variation includes the permitted one extra row. The MobiLink synchronization server automatically stores the user name in the first column.

```
SELECT user_name, order_id, cust_id, prod_id,  
       emp_id, disc, quant, notes, status  
FROM ULOldOrder
```

Java example

This script must return valid SQL.

The following stored procedure call registers a Java method called `oldRowCursor` as the script for the `old_row_cursor` event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(  
    'ver1',  
    'table1',  
    'old_row_cursor',  
    'ExamplePackage.ExampleClass.oldRowCursor' )
```

Following is the sample Java method `oldRowCursor`. It dynamically generates an old row cursor statement by calling a Java method.

```
public String oldRowCursor()  
{ return( getRowCursor( _curTable ) ); }
```

prepare_for_download connection event

Function Processes any required operations between the upload and download transactions.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. You must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	last_download	TIMESTAMP
2	ml_username	VARCHAR(128)

Default action None.

Description The MobiLink synchronization server executes this script as a separate transaction, between the upload transaction and the start of the download transaction.

The last_download timestamp is the value obtained from the consolidated database during the last successful synchronization immediately prior to the download phase. If the current user has never synchronized successfully, this value is set to 1900-01-01.

See also [“end_upload connection event” on page 162](#)
[“begin_download connection event” on page 110](#)

Java example The following stored procedure call registers a Java method called prepareForDownload as the script for the prepare_for_download event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_connection_script(  
    'ver1',  
    'prepare_for_download',  
    'ExamplePackage.ExampleClass.prepareForDownload' )
```

Following is the sample Java method prepareForDownload. It calls a Java method to modify some rows in the database.

```
public String prepareForDownload( Timestamp ts,
String user )
{   adjustUploadedRows( _syncConn, user );
    return( null ); }
```

.NET example

The following stored procedure call registers a .NET method called PrepareForDownload as the script for the prepare_for_download connection event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_connection_script( 'ver1',
'prepare_for_download',
'TestScripts.Test.PrepareForDownload'
)
```

Following is the C# signature for the call PrepareForDownload.

```
public void PrepareForDownload(
    DateTime timestamp,
    string user )
```

report_error connection event

Function Allows you to log errors and to record the actions selected by the handle_error script.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [MobiLink Synchronization User’s Guide, page 233] and “SQL-.NET data types” [MobiLink Synchronization User’s Guide, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	action_code	INTEGER. This parameter is mandatory.
2	error_code	INTEGER. This parameter is optional if none of the following parameters are specified.
3	error_message	TEXT. This parameter is optional if none of the following parameters are specified.
4	ml_username	VARCHAR(128). This parameter is optional if none of the following parameters are specified.
5	table	VARCHAR(128). This parameter is optional.

Default action None.

Description This script allows you to log errors and to record the actions selected by the handle_error script. This script is executed after the handle_error event, whether or not a handle_error script is defined. It is always executed in its own transaction, on a different database connection than the synchronization connection (the administrative/information connection).

The error code and error message allow you to identify the nature of the error. The action code value is returned by the last call to an error handling script for the SQL operation that caused the current error.

If the error happened as part of synchronization, the user name is supplied. Otherwise, this value is NULL.

If the error happened while manipulating a particular table, the table name is

supplied. Otherwise, this value is NULL. The table name is the name of a table in the remote database. This name may or may not have a direct counterpart in the consolidated database, depending on the design of the synchronization system.

See also

[“handle_error connection event” on page 174](#)

[“handle_odbc_error connection event” on page 177](#)

[“report_odbc_error connection event” on page 196](#)

SQL example

The following example works with an Adaptive Server Anywhere consolidated database. It inserts a row into a table used to record synchronization errors.

```
call ml_add_connection_script(
  'ver1',
  'report_error',
  'insert into sync_error(
    action_code,
    error_code,
    error_message,
    user_name,
    table_name )
    values( ?, ?, ?, ?, ? )' )
```

Java example

The following stored procedure call registers a Java method called `reportError` as the script for the `report_error` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_connection_script(
  'ver1',
  'report_error',
  'ExamplePackage.ExampleClass.reportError' )
```

Following is the sample Java method `reportError`. It logs the error to a table using the JDBC connection provided by `MobiLink`. It also sets the action code.

```
public String reportError(
  ianywhere.ml.script.InOutInteger actionCode,
  int errorCode, String errorMessage, String user,
  String table )
  throws java.sql.SQLException
{ // insert error information in a table
  JDBCLogError( _syncConn, errorCode, errorMessage,
    user, table );
  actionCode.setValue( getActionCode( errorCode ) );
  return( null ); }
```

report_odbc_error connection event

Function Allows you to log errors and to record the actions selected by the handle_odbc_error script.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [MobiLink Synchronization User’s Guide, page 233] and “SQL-.NET data types” [MobiLink Synchronization User’s Guide, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	action_code	INTEGER. This parameter is mandatory.
2	ODBC_state	VARCHAR(5). This parameter is optional if none of the following parameters are specified.
3	error_message	TEXT. This parameter is optional if none of the following parameters are specified.
4	ml_username	VARCHAR(128). This parameter is optional if none of the following parameters are specified.
5	table	VARCHAR(128). This parameter is optional.

Default action None.

Description This script allows you to log errors and to record the actions selected by the handle_odbc_error script. This script is executed after the handle_odbc_error event, whether or not a handle_odbc_error script is defined. It is always executed in its own transaction, on a different database connection than the synchronization connection (the administrative/information connection).

The error code and error message allow you to identify the nature of the error. The action code value is returned by the last call to an error handling script for the SQL operation that caused the current error.

If the error happened as part of synchronization, the user name is supplied.

Otherwise, this value is NULL.

If the error happened while manipulating a particular table, the table name is supplied. Otherwise, this value is NULL. The table name is the name of a table in the remote database. This name may or may not have a direct counterpart in the consolidated database, depending on the design of the synchronization system.

See also

[“handle_error connection event” on page 174](#)

[“handle_odbc_error connection event” on page 177](#)

[“report_error connection event” on page 194](#)

SQL example

The following example works with an Adaptive Server Anywhere consolidated database. It inserts a row into a table used to record synchronization errors.

```
call ml_add_connection_script(
  'ver1',
  'report_odbc_error',
  'insert into sync_error(
    action_code,
    odbc_state,
    error_message,
    user_name,
    table_name )
  values( ?, ?, ?, ?, ? )' )
```

Java example

The following stored procedure call registers a Java method called `reportODBCError` as the script for the `report_odbc_error` event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_connection_script(
  'ver1',
  'report_odbc_error',
  'ExamplePackage.ExampleClass.reportODBCError' )
```

Following is the sample Java method `reportODBCError`. It logs the error to a table using the JDBC connection provided by MobiLink. It also sets the action code.

```
public String reportODBCError(
    ianywhere.ml.script.InOutInteger actionCode,
    String ODBCState,
    String errorMessage,
    String user,
    String table )
    throws java.sql.SQLException
{
    JDBCLogError( _syncConn, ODBCState, errorMessage,
        user, table );
    actionCode.setValue( getActionCode( ODBCState ) );
    return( null ); }
```

resolve_conflict table event

Function

Defines a process for resolving a conflict in a specific table.

Parameters

In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. You must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	ml_username	VARCHAR(128)
2	table	VARCHAR(128)

Default action

None.

Description

When a row is updated on a remote database, the MobiLink client saves a copy of the original values. The client sends both old and new values to the MobiLink synchronization server.

When the MobiLink synchronization server receives an updated row, it compares the original values with the present values in the consolidated database. The comparison is carried out using the `upload_fetch` script or, if using cursor-based uploads, the `upload_cursor` script.

If the old uploaded values do not match the current values in the consolidated database, the row conflicts. Instead of updating the row, the MobiLink synchronization server inserts both old and new values into the consolidated database. The old and new rows are handled using the `upload_old_row_insert` and `upload_new_row_insert` scripts, respectively. If you are using cursor-based uploads the rows are handled using `old_row_cursor` and `new_row_cursor`, respectively.

Once the values have been inserted, the MobiLink synchronization server executes the `resolve_conflict` script. It provides the opportunity to resolve the conflict. You can implement any scheme of your choosing.

This script is executed once per conflict.

Alternatively, instead of defining the `resolve_conflict` script, you can resolve conflicts in a set-oriented fashion by putting conflict-resolution logic either in your `end_upload_rows` script or in your `end_upload` table script.

You can have one `resolve_conflict` script for each table in the remote

database.

See also

[“upload_old_row_insert table event” on page 222](#)

[“upload_new_row_insert table event” on page 220](#)

[“upload_update table event” on page 231](#)

[“old_row_cursor cursor event \(deprecated\)” on page 189](#)

[“new_row_cursor cursor event \(deprecated\)” on page 186](#)

[“end_upload_rows table event” on page 168](#)

SQL example

The following statement defines a *resolve_conflict* script suited to the CustDB sample application for an Oracle installation. It calls a stored procedure *ULResolveOrderConflict* .

```
exec ml_add_table_script(
    'custdb', 'ULOrder', 'resolve_conflict',
    'begin ULResolveOrderConflict();
end; ')
CREATE OR REPLACE PROCEDURE ULResolveOrderConflict()
AS
    new_order_id integer;
    new_status   varchar(20);
    new_notes    varchar(50);
BEGIN
    -- approval overrides denial
    SELECT order_id, status, notes
        INTO new_order_id, new_status, new_notes
        FROM ULNewOrder
        WHERE syncuser_id = SyncUserID;
    IF new_status = 'Approved' THEN
        UPDATE ULOrder o
            SET o.status = new_status, o.notes =
                new_notes
            WHERE o.order_id = new_order_id;
    END IF;
    DELETE FROM ULOldOrder;
    DELETE FROM ULNewOrder;
END;
```

Java example

The following stored procedure call registers a Java method called *resolveConflict* as the script for the *resolve_conflict* table event when synchronizing the script version *ver1*. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(
    'ver1',
    'table1',
    'resolve_conflict',
    'ExamplePackage.ExampleClass.resolveConflict' )
```

Following is the sample Java method `resolveConflict`. It calls a Java method that will use the JDBC connection provided by MobiLink. It also sets the action code.

```
public String resolveConflict( String user,
    String table)
{   resolveRows(_syncConn, user ); }
```

synchronization_statistics connection event

Function Tracks synchronization statistics.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	ml_username	VARCHAR(128)
2	warnings	INTEGER
3	errors	INTEGER
4	deadlocks	INTEGER
5	synchronized_tables	INTEGER
6	connection_retries	INTEGER

Default action None.

Description The synchronization_statistics event allows you to gather, for any user and connection, various statistics about the current synchronization. The synchronization_statistics connection script is called just prior to the commit at the end of the end synchronization transaction.

See also [“download_statistics connection event” on page 139](#)
[“download_statistics table event” on page 142](#)
[“upload_statistics connection event” on page 224](#)
[“upload_statistics table event” on page 227](#)
[“synchronization_statistics table event” on page 205](#)
[“time_statistics connection event” on page 207](#)
[“time_statistics table event” on page 209](#)
[“MobiLink Monitor”](#) [*MobiLink Synchronization User’s Guide*, page 297]

SQL example The following example comes from an Oracle installation.


```
INSERT INTO sync_con_audit(
    id, ml_user, warnings, errors,
    deadlocks, synchronized_tables,
    connection_retries)
VALUES (s_audit.nextval,?,?,?,?,?,?)
```

Once statistics are inserted into the audit table, you may use these statistics to monitor your synchronizations and make optimizations where applicable.

Java example

The following stored procedure call registers a Java method called `synchronizationStatisticsConnection` as the script for the `synchronization_statistics` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_connection_script(
    'ver1',
    'synchronization_statistics',
    'ExamplePackage.ExampleClass.synchronizationStatisticsConnecti
on'
)
```

Following is the sample Java method `synchronizationStatisticsConnection`. It logs some of the statistics to the MobiLink output log. (This might be useful at development time but would slow down a production server.)

```
public String synchronizationStatisticsConnection(
    String user, int warnings, int errors, int deadlocks,
    int synchronizedTables, int connectionRetries )
{
    java.lang.System.out.println( "synch statistics
number of deadlocks: " + deadlocks ;
    return( null ); }
}
```

.NET example

The following stored procedure call registers a .NET method called `SyncStats` as the script for the `synchronization_statistics` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_connection_script(
    'ver1',
    'synchronization_statistics',
    'TestScripts.Test.SyncStats'
)
```

Following is the C# signature for the call `SyncStats`.

```
public void SyncStats(  
    string user,  
    int warnings,  
    int errors,  
    int deadLocks,  
    int syncedTables,  
    int connRetries )
```

synchronization_statistics table event

Function Tracks synchronization statistics.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	ml_username	VARCHAR(128)
2	table	VARCHAR(128)
3	warnings	INTEGER
4	errors	INTEGER

Default action None.

Description The synchronization_statistics event allows you to gather, for any user and table, the number of warnings and errors that occurred during synchronization. The synchronization_statistics table script is called just prior to the commit at the end of the end synchronization transaction.

See also [“download_statistics connection event” on page 139](#)
[“download_statistics table event” on page 142](#)
[“upload_statistics connection event” on page 224](#)
[“upload_statistics table event” on page 227](#)
[“synchronization_statistics connection event” on page 202](#)
[“time_statistics connection event” on page 207](#)
[“time_statistics table event” on page 209](#)
[“MobiLink Monitor” \[*MobiLink Synchronization User’s Guide*, page 297\]](#)

SQL example The following example comes from an Oracle installation.

```
INSERT INTO sync_tab_audit (id, ml_user, table,
warnings, errors) VALUES (s_audit.nextval,?,?,?,?)
```

Once synchronization statistics are inserted into the audit table, you may use

these statistics to monitor your synchronizations and make optimizations where applicable.

Java example

The following stored procedure call registers a Java method called `synchronizationStatisticsTable` as the script for the `synchronization_statistics` table event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(  
    'ver1',  
    'table1',  
    'synchronization_statistics',  
    'ExamplePackage.ExampleClass.synchronizationStatisticsTable'  
)
```

Following is the sample Java method `synchronizationStatisticsTable`. It logs some of the statistics to the MobiLink output log. (This might be useful at development time but would slow down a production server.)

```
public String synchronizationStatisticsTable(  
    String user, String table, int warnings, int errors )  
{    java.lang.System.out.println( "synch statistics for  
    table: " + table + " errors: " + errors );  
    return( null ); }
```

.NET example

The following stored procedure call registers a .NET method called `SyncTableStats` as the script for the `synchronization_statistics` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script(  
    'ver1',  
    'table1',  
    'synchronization_statistics',  
    'TestScripts.Test.SyncTableStats'  
)
```

Following is the C# signature for the call `SyncTableStats`.

```
public void SyncTableStats(  
    string user,  
    string table,  
    int warnings,  
    int errors )
```

time_statistics connection event

Function Tracks time statistics by user and event.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	ml_username	VARCHAR(128)
2	event_name	VARCHAR(128)
3	num_calls	INTEGER
4	min_time	INTEGER. Milliseconds.
5	max_time	INTEGER. Milliseconds.
6	total_time	INTEGER. Milliseconds.

Default action None.

Description The time_statistics event allows you to gather time statistics for any user during synchronization. The statistics are gathered only for those events for which there is a corresponding script. The script gathers aggregate data for occasions where a single event occurs multiple times. The script can be especially useful for time comparisons across users, events and tables.

See also [“time_statistics table event” on page 209](#)
[“download_statistics connection event” on page 139](#)
[“download_statistics table event” on page 142](#)
[“upload_statistics connection event” on page 224](#)
[“upload_statistics table event” on page 227](#)
[“synchronization_statistics connection event” on page 202](#)
[“synchronization_statistics table event” on page 205](#)
[“MobiLink Monitor”](#) [*MobiLink Synchronization User’s Guide*, page 297]

SQL example The following example comes from an Oracle installation.

```
INSERT INTO time_statistics (id, ml_user, table,
    event_name, num_calls, min_time, max_time, total_time)
VALUES (ts_id.nextval,?,?,?, ?,?,?)
```

Java example

The following stored procedure call registers a Java method called `timeStatisticsConnection` as the script for the `time_statistics` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_connection_script(
    'ver1',
    'time_statistics',
    'ExamplePackage.ExampleClass.timeStatisticsConnection' )
```

Following is the sample Java method `timeStatisticsConnection`. It prints statistics for the `prepare_for_download` event.

```
public void timeStatisticsConnection(
    String ml_username,
    String table_name,
    String event_name,
    int num_calls, int min_time, int max_time,
    int total_time )
{ if( event_name.equals( "prepare_for_download" )
    { java.lang.System.out.println(
        "prepare_for_download num_calls: " + num_calls +
        "total_time: " + total_time ); } }
```

.NET example

The following stored procedure call registers a .NET method called `TimeStats` as the script for the `time_statistics` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_connection_script(
    'ver1',
    'time_statistics',
    'TestScripts.Test.TimeStats'
)
```

Following is the C# signature for the call `TimeStats`.

```
public void TimeStats(
    string user,
    string eventName,
    int numCalls,
    int minTime,
    int maxTime,
    int totTime )
```

time_statistics table event

Function Tracks time statistics.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	ml_username	VARCHAR(128)
2	table	VARCHAR(128)
3	event_name	VARCHAR(128)
4	num_calls	INTEGER
5	min_time	INTEGER. Milliseconds.
6	max_time	INTEGER. Milliseconds.
7	total_time	INTEGER. Milliseconds.

Default action None.

Description The time_statistics table event allows you to gather time statistics for any user and table during synchronization. The statistics are gathered only for those events for which there is a corresponding script. The script gathers aggregate data for occasions where a single event occurs multiple times. The script can be especially useful for time comparisons across users, events and tables.

See also [“time_statistics connection event” on page 207](#)
[“download_statistics connection event” on page 139](#)
[“download_statistics table event” on page 142](#)
[“upload_statistics connection event” on page 224](#)
[“upload_statistics table event” on page 227](#)
[“synchronization_statistics connection event” on page 202](#)
[“synchronization_statistics table event” on page 205](#)

“MobiLink Monitor” [MobiLink Synchronization User’s Guide, page 297]

SQL example

The following example comes from an Oracle installation.

```
INSERT INTO time_statistics(  
    id, ml_user, table, event_name, num_calls,  
    min_time, max_time, total_time)  
VALUES (ts_id.nextval,?,?,?,?,?,?,?)
```

Java example

The following stored procedure call registers a Java method called timeStatisticsTable as the script for the time_statistics table event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(  
    'ver1',  
    'table1',  
    'time_statistics',  
    'ExamplePackage.ExampleClass.timeStatisticsTable' )
```

Following is the sample Java method timeStatisticsTable. It prints statistics for the upload_old_row_insert event.

```
public void timeStatisticsConnection(  
    String ml_username,  
    String table_name,  
    String event_name,  
    int num_calls, int min_time, int max_time,  
    int total_time )  
{ if( event_name.equals( "upload_old_row_insert")  
  { java.lang.System.out.println(  
    "upload_old_row_insert num_calls: " + num_calls +  
    "total_time: " + total_time ); } }
```

.NET example

The following stored procedure call registers a .NET method called TimeTableStats as the script for the time_statistics table event when synchronizing the script version ver1 and the table table1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script(  
    'ver1',  
    'table1',  
    'time_statistics',  
    'TestScripts.Test.TimeTableStats'  
)
```

Following is the C# signature for the call TimeTableStats.


```
public void TimeTableStats(  
    string user,  
    string table,  
    string eventName,  
    int numCalls,  
    int minTime,  
    int maxTime,  
    int totTime )
```

upload_cursor cursor event (deprecated)

Function Defines a cursor that the MobiLink synchronization server uses to insert, update, or delete rows during processing of the upload stream.

Use statement-based events for uploads

This script has been deprecated. Use the statement-based events `upload_delete`, `upload_insert`, and `upload_update` instead of the `upload_cursor` event to process the upload stream. Support for the `upload_cursor` event is likely to be removed from future releases.

Item	Parameter
1	primary key 1
2	primary key 2
...	...

Default action None.

Description The MobiLink synchronization server opens a cursor with which to insert, update, or delete rows in the consolidated database based on rows uploaded from a client application. This script should contain a suitable `SELECT` statement or call a stored procedure that contains a suitable `SELECT` statement.

The parameters are the values of each column included in the primary key of the corresponding client table. You must use these in a `WHERE` clause, so that the synchronization can identify a unique row based on these values. The type and order of the parameters is as defined in the `example_upload_cursor` script. This order is the same as that in the corresponding table definition in the remote database, which in turn may have been copied from your reference database.

You can have one `upload_cursor` script for each table in the remote database. For Java and .NET applications, this script must return valid SQL.

See also “Writing scripts to upload rows” [*MobiLink Synchronization User’s Guide*, page 54]
[“upload_delete table event” on page 214](#)
[“upload_insert table event” on page 218](#)
[“upload_update table event” on page 231](#)

SQL example

The following SELECT statement defines the upload cursor in the CustDB sample application.

```
SELECT cust_id, cust_name
FROM ULCustomer
WHERE cust_id = ?
```

The primary key of the ULCustomer table in the CustDB sample application is the column cust_id. If the corresponding table in the consolidated database is, instead, named Customer, then change the above statement as follows.

```
SELECT cust_id, cust_name
FROM Customer
WHERE cust_id = ?
```

Java example

The following stored procedure call registers a Java method called uploadCursor as the script for the upload_cursor cursor event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(
    'ver1',
    'table1',
    'upload_cursor',
    'ExamplePackage.ExampleClass.uploadCursor' )
```

Following is the sample Java method uploadCursor. It dynamically generates an upload cursor.

```
public String uploadCursor()
{ return( getUploadCursor( _curTable ) ); }
```

.NET example

The following C# example deletes the contents of a temporary table. It then returns SQL that causes rows to be uploaded into the temporary table.

```
public string UploadCursor()
{
    DBCommand stmt = curConn.CreateCommand();
    stmt.CommandText = "DELETE FROM dnet_ul_temp";
    stmt.ExecuteNonQuery();
    stmt.Close();

    return( "SELECT * FROM dnet_ul_temp WHERE pk = ?" );
}
```

upload_delete table event

Function Provides an event that the MobiLink synchronization server uses during processing of the upload stream to handle rows deleted from the remote database.

Parameters

Item	Parameter
1	primary key 1
2	primary key 2
...	...

Default action None.

Description The statement-based upload_delete script handles rows that are deleted in the remote database. The action taken at the consolidated database can be a DELETE statement, but need not be.

You can have one upload_delete script for each table in the remote database.

For Java and .NET applications, this script must return valid SQL.

See also [“upload_insert table event” on page 218](#)

[“upload_update table event” on page 231](#)

SQL example This example is taken from the Contact sample and can be found in *Samples\MobiLink\Contact\build_consol.sql*. It marks customers that are deleted from the remote database as inactive.

```
UPDATE Customer SET active = 0 WHERE cust_id=?
```

Java example The following stored procedure call registers a Java method called uploadDeleteTable as the script for the upload_delete table event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(  
    'ver1',  
    'table1',  
    'upload_delete',  
    'ExamplePackage.ExampleClass.uploadDeleteTable' )
```

Following is the sample Java method uploadDeleteTable. It dynamically generates an UPLOAD statement.

```
public string uploadDeleteTable()  
{ return( genUD(_curTable) ); }
```

.NET example

The following stored procedure call registers a .NET method called UploadDelete as the script for the upload_delete table event when synchronizing the script version ver1 and the table table1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script(  
    'ver1',  
    'table1',  
    'upload_delete',  
    'TestScripts.Test.UploadDelete'  
)
```

Following is the C# signature for the call UploadDelete.

```
public string UploadDelete( object pk1 )
```

upload_fetch table event

Function Provides an event that the MobiLink synchronization server uses to identify update conflicts during statement-based processing of the upload stream.

Parameters

Item	Parameter
1	primary key 1
2	primary key 2
...	...

Default action None.

Description The statement-based upload_fetch script fetches rows from a synchronized table for the purposes of conflict detection. It is a companion to the upload_update event.

The columns of the result set must match the number of columns being uploaded from the remote database for this table. If the values returned do not match the before image in the uploaded row, a conflict is identified.

You can have one upload_fetch script for each table in the remote database.

See also [“resolve_conflict table event” on page 199](#)

[“upload_delete table event” on page 214](#)

[“upload_insert table event” on page 218](#)

[“upload_update table event” on page 231](#)

SQL example The following SQL script is taken from the Contact sample and can be found in *Samples\MobiLink\Contact\build_consol.sql*. It is used to identify conflicts that occur when rows updated in the remote database Product table are uploaded. This script selects rows from a table also named Product, but depending on your consolidated and remote database schema, the two table names may not match.

```
SELECT id, name, size, quantity, unit_price
FROM Product WHERE id=?
```

Java example This script must return valid SQL.

The following stored procedure call registers a Java method called uploadFetchTable as the script for the upload_fetch table event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(  
    'ver1',  
    'table1',  
    'upload_fetch',  
    'ExamplePackage.ExampleClass.uploadFetchTable' )
```

Following is the sample Java method `uploadFetchTable`. It dynamically generates an `UPLOAD` statement.

```
public string uploadFetchTable()  
{    return( genUF(_curTable) ); }
```

upload_insert table event

Function Provides an event that the MobiLink synchronization server uses during processing of the upload stream to handle rows inserted into the remote database.

Parameters

Item	Parameter
1	column 1
2	column 2
...	...

Default action None.

Description The statement based upload_insert script performs direct inserts of column values.

You can have one upload_insert script for each table in the remote database.

For Java and .NET applications, this script must return valid SQL.

See also [“upload_delete table event” on page 214](#)

[“upload_update table event” on page 231](#)

[“upload_fetch table event” on page 216](#)

SQL example This example is taken from the Contact sample and can be found in *Samples\MobiLink\Contact\build_consol.sql*. It handles inserts made on the Customer table in the remote database. The script inserts the values into a table named Customer in the consolidated database. The final column of the table identifies the Customer as active. The final column does not appear in the remote database.

```
INSERT INTO Customer( cust_id, name, rep_id, active )
VALUES ( ?, ?, ?, 1 )
```

Java example The following stored procedure call registers a Java method called uploadInsertTable as the script for the upload_insert table event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(
    'ver1',
    'table1',
    'upload_insert',
    'ExamplePackage.ExampleClass.uploadInsertTable' )
```


Following is the sample Java method `uploadInsertTable`. It dynamically generates an `UPLOAD` statement.

```
public string uploadInsertTable()  
{ return("insert into" + _curTable + getCols(_curTable)  
  + "values" + getQM(_curTable)); }
```

.NET example

The following stored procedure call registers a .NET method called `UploadInsert` as the script for the `upload_insert` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script(  
  'ver1',  
  'table1',  
  'upload_insert',  
  'TestScripts.Test.UploadInsert'  
)
```

Following is the C# signature for the call `UploadInsert`.

```
public string UploadInsert( string user )
```

upload_new_row_insert table event

Function Conflict resolution scripts for statement-based uploads commonly require access to the old and new values of rows updated in the remote database. This event allows you to handle the new values of rows updated rows in the remote database during conflict resolution.

Parameters

Item	Parameter
1	column 1
2	column 2
...	...

Default action None.

Description You can use this event to assist in developing conflict resolution procedures for statement-based updates. The event parameters hold the values for the row in the remote database before the update was carried out. It is also used to insert `INSERTed` rows in statement-based, forced-conflict mode.

A typical action for this event is to hold the row in a temporary table for use by a `resolve_conflict` script.

You can have one `upload_new_row_insert` script for each table in the remote database.

For Java and .NET applications, this script must return valid SQL.

See also “Handling conflicts” [*MobiLink Synchronization User’s Guide*, page 90]

- “[resolve_conflict table event](#)” on page 199
- “[upload_old_row_insert table event](#)” on page 222
- “[upload_update table event](#)” on page 231

SQL example This example is taken from the Contact sample and can be found in `Samples\MobiLink>Contact\build_consol.sql`. It handles updates made on the product table in the remote database. The script inserts the new value of the row into a global temporary table named `product_conflict`. The final column of the table identifies the row as a new row.

```
INSERT INTO DBA.product_conflict(  
    id, name, size, quantity, unit_price, row_type )  
VALUES( ?, ?, ?, ?, ?, 'N' )
```

Java example The following stored procedure call registers a Java method called `uploadNewRowInsertTable` as the script for the `upload_new_row_insert`

table event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(  
    'ver1',  
    'table1',  
    'upload_new_row_insert',  
    'ExamplePackage.ExampleClass.uploadNewRowInsertTable'  
)
```

Following is the sample Java method `uploadNewRowInsertTable`. It dynamically generates an `UPLOAD` statement.

```
public string uploadNewRowInsertTable()  
{ return("insert into" + _curTable + "_new" +  
    getCols(_curTable) + "values" + getQM(_curTable)); }
```

upload_old_row_insert table event

Function Conflict resolution scripts for statement-based uploads commonly require access to the old and new values of rows updated in the remote database. This event allows you to handle the new values of rows that were updated in the remote database during conflict resolution.

Parameters

Item	Parameter
1	column 1
2	column 2
...	...

Default action None.

Description The statement based upload_old_row_insert script performs direct insert of column values as specified in the upload_old_row_insert statement. You can have one upload_old_row_insert script for each table in the remote database.

See also “Handling conflicts” [*MobiLink Synchronization User’s Guide*, page 90]
[“resolve_conflict table event” on page 199](#)
[“upload_new_row_insert table event” on page 220](#)
[“upload_update table event” on page 231](#)

SQL example This example is taken from the Contact sample and can be found in *Samples\MobiLink>Contact\build_consol.sql*. It handles updates made on the product table in the remote database. The script inserts the old value of the row into a global temporary table named product_conflict. The final column of the table identifies the row as an old row.

```
insert into DBA.product_conflict(
id, name, size, quantity, unit_price, row_type )
values( ?, ?, ?, ?, ?, 'O' )
```

Java example The following stored procedure call registers a Java method called uploadOldRowInsertTable as the script for the upload_old_row_insert table event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(  
    'ver1',  
    'table1',  
    'upload_old_row_insert',  
    'ExamplePackage.ExampleClass.uploadNewRowInsertTable'  
)
```

Following is the sample Java method `uploadOldRowInsertTable`. It dynamically generates an `UPLOAD` statement.

```
public string uploadOldRowInsertTable()  
{    old" + getCols(_curTable) +  
    "values" + getQM(_curTable)); }
```

upload_statistics connection event

Function Tracks synchronization statistics for upload operations.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	ml_username	VARCHAR(128)
2	warnings	INTEGER
3	errors	INTEGER
4	inserted_rows	INTEGER
5	deleted_rows	INTEGER
6	updated_rows	INTEGER
7	conflicted_inserts	INTEGER
8	conflicted_deletes	INTEGER
9	conflicted_updates	INTEGER
10	ignored_inserts	INTEGER
11	ignored_deletes	INTEGER
12	ignored_updates	INTEGER
13	bytes	INTEGER
14	deadlocks	INTEGER

Default action None.

Description The upload_statistics event allows you to gather, for any user, statistics on uploads. The upload_statistics connection script is called just prior to the commit at the end of the upload transaction.

See also [“download_statistics connection event” on page 139](#)

[“download_statistics table event” on page 142](#)

[“upload_statistics table event” on page 227](#)

[“synchronization_statistics connection event” on page 202](#)

[“synchronization_statistics table event” on page 205](#)

[“time_statistics connection event” on page 207](#)

[“time_statistics table event” on page 209](#)

[“MobiLink Monitor” \[MobiLink Synchronization User’s Guide, page 297\]](#)

SQL example

The following example comes from an Oracle installation.

```
INSERT INTO upload_summary_audit (
  id, ml_user, warnings, errors, inserted_rows,
  deleted_rows, updated_rows, conflicted_inserts,
  conflicted_deletes, conflicted_updates,
  bytes, ignored_inserts, ignored_deletes,
  ignored_updates, bytes, deadlocks)
VALUES (usa_audit.nextval,?,?,?,?,?,?,?,?,?,?,?,?,?)
```

Once statistics are inserted into the audit table, you may use these statistics to monitor your synchronizations and make optimizations where applicable.

Java example

The following stored procedure call registers a Java method called `uploadStatisticsConnection` as the script for the `upload_statistics` connection event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_connection_script(
  'ver1',
  'upload_statistics',
  'ExamplePackage.ExampleClass.uploadStatisticsConnection' )
```

Following is the sample Java method `uploadStatisticsConnection`. It logs some statistics to the MobiLink output log. (This might be useful at development time but would slow down a production server.)

```
public String uploadStatisticsConnection(
  String user,
  int warnings,
  int errors,
  int insertedRows,
  int deletedRows,
  int updatedRows,
  int conflictedInserts,
  int conflictedDeletes,
```

```
        int conflictedUpdates,
        int ignoredInserts,
        int ignoredDeletes,
        int ignoredUpdates,
        int bytes,
        int deadlocks
    )
    {   java.lang.System.out.println( "updated rows: " +
        updatedRows ); }
```

.NET example

The following stored procedure call registers a .NET method called UploadStats as the script for the upload_statistics connection event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_connection_script(
    'ver1',
    'upload_statistics',
    'TestScripts.Test.UploadStats'
)
```

Following is the C# signature for the call UploadStats.

```
public void UploadStats(
    string user,
    int warnings,
    int errors,

    int insertedRows,
    int deletedRows,
    int updatedRows,
    int conflictInserts,
    int conflictDeletes,
    int conflictUpdates,
    int ignoredInserts,
    int ignoredDeletes,
    int ignoredUpdates,
    int bytes,
    int deadlocks )
```


upload_statistics table event

Function Tracks synchronization statistics for upload operations for a specific table.

Parameters In the following table, the description provides the SQL data type. If you are writing your script in Java or .NET, you should use the appropriate corresponding data type. See “SQL-Java data types” [*MobiLink Synchronization User’s Guide*, page 233] and “SQL-.NET data types” [*MobiLink Synchronization User’s Guide*, page 261].

Event parameters are optional only if no subsequent parameters are specified. For example, you must use parameter 1 if you want to use parameter 2.

Item	Parameter	Description
1	ml_username	VARCHAR(128)
2	table	VARCHAR(128)
3	warnings	INTEGER
4	errors	INTEGER
5	inserted_rows	INTEGER
6	deleted_rows	INTEGER
7	updated_rows	INTEGER
8	conflicted_inserts	INTEGER
9	conflicted_deletes	INTEGER
10	conflicted_updates	INTEGER
11	ignored_inserts	INTEGER
12	ignored_deletes	INTEGER
13	ignored_updates	INTEGER
14	bytes	INTEGER
15	deadlocks	INTEGER

Default action None.

Description The upload_statistics event allows you to gather, for any user, vital statistics on synchronization happenings as they apply to any table. The upload_statistics table script is called just prior to the commit at the end of

the upload transaction.

See also

[“download_statistics connection event” on page 139](#)

[“upload_statistics connection event” on page 224](#)

[“upload_statistics table event” on page 227](#)

[“synchronization_statistics connection event” on page 202](#)

[“synchronization_statistics table event” on page 205](#)

[“time_statistics connection event” on page 207](#)

[“time_statistics table event” on page 209](#)

[“MobiLink Monitor” \[MobiLink Synchronization User’s Guide, page 297\]](#)

SQL Example

The following example works with an Adaptive Server Anywhere consolidated database. It inserts a row into a table used to track upload statistics.

```
call ml_add_connection_script(
  'ver1',
  'upload_statistics',
  'insert into my_upload_statistics
    ( user_name, table_name, num_warnings, num_errors,
      inserted_rows, deleted_rows, updated_rows,
      conflicted_inserts, conflicted_deletes, conflicted_
        updates,
      ignored_inserts, ignored_deletes, ignored_updates,
      bytes, deadlocks )
  values( ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ? )' )
```

The following example works with an Oracle consolidated database.

```
INSERT INTO upload_tables_audit (
  id, user_name, table, warnings, errors,
  inserted_rows, deleted_rows, updated_rows,
  conflicted_inserts, conflicted_deletes,
  conflicted_updates, ignored_inserts, ignored_deletes,
  ignored_updates, bytes, deadlocks)
VALUES ( ut_audit.nextval,
  ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
```

Once statistics are inserted into the audit table, you may use these statistics to monitor your synchronizations and make optimizations where applicable.

Java example

The following stored procedure call registers a Java method called `uploadStatisticsTable` as the script for the `upload_statistics` table event when synchronizing the script version `ver1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(  
    'ver1',  
    'table1',  
    'upload_statistics',  
    'ExamplePackage.ExampleClass.uploadStatisticsTable' )
```

Following is the sample Java method `uploadStatisticsTable`. It logs some statistics to the MobiLink output log. (This might be useful at development time but would slow down a production server.)

```
public String uploadStatisticsTable(  
    String user,  
    int warnings,  
    int errors,  
    int insertedRows,  
    int deletedRows,  
    int updatedRows,  
    int conflictedInserts,  
    int conflictedDeletes,  
    int conflictedUpdates,  
    int ignoredInserts,  
    int ignoredDeletes,  
    int ignoredUpdates,  
    int bytes,  
    int deadlocks  
)  
{  
    java.lang.System.out.println( "updated rows: " +  
        updatedRows );  
}
```

.NET example

The following stored procedure call registers a .NET method called `UploadTableStats` as the script for the `upload_statistics` table event when synchronizing the script version `ver1` and the table `table1`. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script(  
    'ver1',  
    'table1',  
    'upload_statistics',  
    'TestScripts.Test.UploadTableStats'  
)
```

Following is the C# signature for the call `UploadTableStats`.

```
public void UploadTableStats(  
    string user,  
    string table,  
    int warnings,  
    int errors,  
    int insertedRows,  
    int deletedRows,  
    int updatedRows,  
    int conflictInserts,  
    int conflictDeletes,  
    int conflictUpdates,  
    int ignoredInserts,  
    int ignoredDeletes,  
    int ignoredUpdates,  
    int bytes,  
    int deadlocks )
```

upload_update table event

Function Provides an event that the MobiLink synchronization server uses during processing of the upload stream to handle rows updated at the remote database.

Parameters

Clause	Parameters
SET	column 11 column 22 ...
WHERE	primary key 1 primary key 2 ...

Default action None.

Description The statement-based upload_update script performs direct update of column values as specified in the upload_update statement.

The WHERE clause must include all of the primary key columns that are being synchronized. The SET clause must contain all of the non-primary key columns that are being synchronized.

You use as many non-primary key columns in your SET clause as exist in the table, and MobiLink will send the correct number of column values. Similarly, in the WHERE clause, you can have any number of primary keys, but all must be specified here, and MobiLink will send the correct values. MobiLink sends these column values and primary key values in the order the columns or primary keys appear in a MobiLink report of your schema. You can use the -vh option to determine the column ordering for this table schema.

You can have one upload_update script for each table in the remote database.

For Java and .NET applications, this script must return valid SQL.

See also [“upload_delete table event” on page 214](#)

[“upload_fetch table event” on page 216](#)

[“upload_insert table event” on page 218](#)

SQL example This example is taken from the Contact sample and can be found in *Samples\MobiLink>Contact\build_consol.sql*. It handles updates made to the Customer table in the remote database. The script updates the values in a

table named Customer in the consolidated database.

```
UPDATE Customer SET name=?, rep_id=? WHERE cust_id=?
```

Java example

The following stored procedure call registers a Java method called uploadUpdateTable as the script for the upload_update table event when synchronizing the script version ver1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_java_table_script(  
    'ver1',  
    'table1',  
    'upload_update',  
    'ExamplePackage.ExampleClass.uploadUpdateTable' )
```

Following is the sample Java method uploadUpdateTable. It dynamically generates an UPLOAD statement.

```
public string uploadUpdateTable()  
{ return( genUU(_curTable) ); }
```

.NET example

The following stored procedure call registers a .NET method called UploadUpdate as the script for the upload_update table event when synchronizing the script version ver1 and the table table1. This syntax is for Adaptive Server Anywhere consolidated databases.

```
call ml_add_dnet_table_script(  
    'ver1',  
    'table1',  
    'upload_update',  
    'TestScripts.Test.UploadUpdate'  
)
```

Following is the C# signature for the call UploadUpdate.

```
public string UploadUpdate()
```

CHAPTER 4

SQL Statements

About this chapter

This chapter presents detailed descriptions of SQL statements in alphabetical order.

Contents

Topic:	page
ALTER PUBLICATION statement	234
ALTER SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]	236
ALTER SYNCHRONIZATION USER statement [MobiLink]	238
CREATE PUBLICATION statement	240
CREATE SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]	243
CREATE SYNCHRONIZATION USER statement [MobiLink]	245
DROP PUBLICATION statement	255
DROP SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]	256
DROP SYNCHRONIZATION USER statement [MobiLink]	257
START SYNCHRONIZATION DELETE statement [MobiLink]	258
STOP SYNCHRONIZATION DELETE statement [MobiLink]	260

ALTER PUBLICATION statement

Description	Use this statement to alter a publication. In MobiLink, a publication identifies synchronized data in a Adaptive Server Anywhere remote database. In SQL Remote, publications identify replicated data in both consolidated and remote databases.
Syntax	ALTER PUBLICATION [<i>owner.</i>] <i>publication-name</i> <i>alterpub-clause</i> , ... <i>alterpub-clause</i> : ADD TABLE <i>article-description</i> MODIFY TABLE <i>article-description</i> { DELETE DROP } TABLE [<i>owner.</i>] <i>table-name</i> RENAME <i>publication-name</i> <i>owner</i> , <i>publication-name</i> , <i>table-name</i> : <i>identifier</i> <i>article-description</i> : <i>table-name</i> [(<i>column-name</i> , ...)] [WHERE <i>search-condition</i>] [SUBSCRIBE BY <i>expression</i>]
Usage	<p>This statement is applicable only to MobiLink and SQL Remote.</p> <p>The ALTER PUBLICATION statement alters a publication in the database. The contribution to a publication from one table is called an article. Changes can be made to a publication by adding, modifying, or deleting articles, or by renaming the publication. If an article is modified, the entire specification of the modified article must be entered.</p> <p>You set options for a MobiLink publication with the ADD OPTION clause in the ALTER SYNCHRONIZATION SUBSCRIPTION statement or CREATE SYNCHRONIZATION SUBSCRIPTION statement.</p>
Permissions	Must have DBA authority, or be the owner of the publication. Requires exclusive access to all tables referred to in the statement.
Side effects	Automatic commit.
See also	<p>“CREATE PUBLICATION statement” on page 240</p> <p>“DROP PUBLICATION statement” on page 255</p> <p>“ALTER SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]” on page 236</p> <p>“CREATE SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]” on page 243</p> <p>“sp_add_article procedure” [SQL Remote User’s Guide, page 379]</p>

“sp_add_article_col procedure” [*SQL Remote User's Guide*, page 381]

Standards and
compatibility

- ◆ **SQL/92** Vendor extension.
- ◆ **SQL/99** Vendor extension.

Example

The following statement adds the customer table to the pub_contact publication.

```
ALTER PUBLICATION pub_contact  
ADD TABLE customer
```

ALTER SYNCHRONIZATION SUBSCRIPTION

statement [MobiLink]

Description Use this statement in an Adaptive Server Anywhere remote database to alter the properties of a subscription of a MobiLink user to a publication.

Syntax

```
ALTER SYNCHRONIZATION SUBSCRIPTION  
TO publication-name  
[ FOR ml_username, ... ]  
[ TYPE sync-type ]  
[ ADDRESS network-parameters ]  
[ ADD OPTION option=value, ... ]  
[ MODIFY OPTION option=value, ... ]  
[ DELETE { ALL OPTION | OPTION option, ... }]
```

ml_username: *identifier*

network-parameters: *string*

sync-type: **http** | **https** | **tcip** | **ActiveSync**

value: *string* | *integer*

Parameters

TO clause Specify the name of a publication.

FOR clause Specify one or more MobiLink user IDs.

Omit the FOR clause to set extended options, sync type and network parameters for a publication.

☞ For information about how dbmlsync processes options that are specified in different locations, see “Priority order for extended options and connection parameters” [*MobiLink Synchronization User’s Guide*, page 180].

TYPE clause This clause specifies the communication protocol to use for synchronization. The default protocol is **tcip**.

ADDRESS clause This clause specifies network parameters, including the location of the MobiLink synchronization server.

☞ For a complete list of network parameters, see “[CREATE SYNCHRONIZATION USER statement \[MobiLink\]](#)” on page 245.

ADD OPTION, MODIFY OPTION, DELETE OPTION AND DELETE ALL OPTION clause These clauses allow you to add, modify, delete or delete all options. You may specify only one parameter in each clause.

The values for each option cannot contain the characters “=” or “,” or “;”.

☞ For a complete list of options, see “[CREATE SYNCHRONIZATION USER statement \[MobiLink\]](#)” on page 245.

Usage	Use this statement to alter a synchronization subscription within a MobiLink remote or reference database.
Permissions	Must have DBA authority. Requires exclusive access to all tables referred to in the publication.
Side effects	Automatic commit.
See also	“CREATE PUBLICATION statement” on page 240 “CREATE SYNCHRONIZATION USER statement [MobiLink]” on page 245
Standards and compatibility	<p>◆ SQL/92 Vendor extension.</p> <p>◆ SQL/99 Vendor extension.</p>
Examples	<p>Create a default subscription, which contains default subscription values, for the sales publication (by omitting the FOR clause). Indicate the address of the MobiLink synchronization server and specify that only the Certicom root certificate is to be trusted.</p>

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  ADDRESS 'host=test.internal;port=2439;
         security=ecc_tls'
  OPTION memory='2m';
```

Subscribe MobiLink user ml_user1 to the sales publication. Set the memory option to 3 Mb, rather than the value specified in the default publication.

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR 'ml_user1'
  OPTION memory='3m';
```

ALTER SYNCHRONIZATION USER statement [MobiLink]

Description	Use this statement in an Adaptive Server Anywhere remote database to alter the properties of a MobiLink user.
Syntax	<pre>ALTER SYNCHRONIZATION USER <i>ml_username</i> [TYPE <i>sync-type</i>] [ADDRESS <i>network-parameters</i>] [ADD OPTION <i>option=value, ...</i>] [MODIFY OPTION <i>option=value, ...</i>] [DELETE { ALL OPTION OPTION <i>option</i> }]</pre> <p><i>ml_username</i>: <i>identifier</i></p> <p><i>network-parameters</i>: <i>string</i></p> <p><i>sync-type</i>: http https tcpip ActiveSync</p> <p><i>value</i>: <i>string</i> <i>integer</i></p>
Parameters	<p>TYPE clause This clause specifies the communication protocol to use for synchronization.</p> <p>ADDRESS clause This clause specifies network parameters, including the location of the MobiLink synchronization server.</p> <p>☞ For a complete list of network parameters, see “CREATE SYNCHRONIZATION USER statement [MobiLink]” on page 245.</p> <p>ADD OPTION, MODIFY OPTION, DELETE OPTION AND DELETE ALL OPTION clause These clauses allow you to add, modify, delete or delete all options. You may specify only one parameter in each clause.</p> <p>☞ For a complete list of options, see “CREATE SYNCHRONIZATION USER statement [MobiLink]” on page 245.</p>
Usage	Use this statement to alter the properties of a synchronization user within a MobiLink remote database.
Permissions	Must have DBA authority. Requires exclusive access to all tables referred to in the publication.
Side effects	Automatic commit.
See also	“ALTER SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]” on page 236
	“CREATE SYNCHRONIZATION USER statement [MobiLink]” on

[page 245](#)

[“CREATE SYNCHRONIZATION SUBSCRIPTION statement \[MobiLink\]”
on page 243](#)

Standards and
compatibility

- ◆ **SQL/92** Vendor extension.
- ◆ **SQL/99** Vendor extension.

CREATE PUBLICATION statement

Description	Use this statement to create a publication. In MobiLink, a publication identifies synchronized data in UltraLite or Adaptive Server Anywhere remote databases. In SQL Remote, publications identify replicated data in both consolidated and remote databases.
Syntax	CREATE PUBLICATION [<i>owner.</i>] <i>publication-name</i> (TABLE <i>article-description</i> , ...) <i>owner</i> , <i>publication-name</i> : <i>identifier</i> <i>article-description</i> : <i>table-name</i> [(<i>column-name</i> , ...)] [WHERE <i>search-condition</i>] [SUBSCRIBE BY <i>expression</i>]
Parameters	<p>article-description Publications are built from articles. Each article is a table or part of a table. An article may be a vertical partition of a table (a subset of the table's columns), a horizontal partition (a subset of the table's rows) or a vertical and horizontal partition.</p> <p>WHERE clause The WHERE clause is a way of defining the subset of rows of a table to be included in an article. It is useful if the same subset is to be received by all subscribers to the publication.</p> <p>SUBSCRIBE BY clause In SQL Remote, one way of defining a subset of rows of a table to be included in an article is to use a SUBSCRIBE BY clause. This clause allows many different subscribers to receive different rows from a table in a single publication definition. This clause is ignored during MobiLink synchronization.</p> <p>You can combine WHERE and SUBSCRIBE BY clauses in an article definition, but the SUBSCRIBE BY clause is used only by SQL Remote.</p>
Usage	<p>This statement is applicable only to MobiLink and SQL Remote.</p> <p>The CREATE PUBLICATION statement creates a publication in the database. A publication can be created for another user by specifying an owner name.</p> <p>In MobiLink, publications are required in Adaptive Server Anywhere remote databases, and are optional in UltraLite databases. These publications and the subscriptions to them determine which data will be uploaded to the MobiLink synchronization server. You can construct a remote database by creating publications and subscriptions directly. Alternatively, you can create publications and subscriptions in an Adaptive Server Anywhere reference database, which acts as a template for the remote databases, and</p>

then construct the remote databases using the MobiLink extraction utility.

You set options for a MobiLink publication with the `ADD OPTION` clause in the `ALTER SYNCHRONIZATION SUBSCRIPTION` statement or `CREATE SYNCHRONIZATION SUBSCRIPTION` statement.

In SQL Remote, publishing is a two-way operation, as data can be entered at both consolidated and remote databases. In a SQL Remote installation, any consolidated database and all remote databases must have the same publication defined. Running the SQL Remote extraction utility from a consolidated database automatically executes the correct `CREATE PUBLICATION` statement in the remote database.

Permissions	Must have DBA authority. Requires exclusive access to all tables referred to in the statement.
Side effects	Automatic commit.
See also	<p>“ALTER PUBLICATION statement” on page 234</p> <p>“DROP PUBLICATION statement” on page 255</p> <p>“ALTER SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]” on page 236</p> <p>“CREATE SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]” on page 243</p> <p>“sp_create_publication procedure” [SQL Remote User’s Guide, page 384]</p>
Standards and compatibility	<p>◆ SQL/92 Vendor extension.</p> <p>◆ SQL/99 Vendor extension.</p>
Example	The following statement publishes all columns and rows of two tables.

```
CREATE PUBLICATION pub_contact (
    TABLE contact,
    TABLE company
)
```

The following statement publishes only some columns of one table.

```
CREATE PUBLICATION pub_customer (
    TABLE customer ( id, company_name, city )
)
```

The following statement publishes only the active customer rows by including a `WHERE` clause that tests the status column of the customer table.

```
CREATE PUBLICATION pub_customer (
    TABLE customer ( id, company_name, city, state )
    WHERE status = 'active'
)
```

The following statement publishes only some rows by providing a subscribe-by value. This method can be used only with SQL Remote.

```
CREATE PUBLICATION pub_customer (  
    TABLE customer ( id, company_name, city, state )  
    SUBSCRIBE BY state  
)
```

The subscribe-by value is used as follows when you create a SQL Remote subscription.

```
CREATE SUBSCRIPTION TO pub_customer ( 'NY' )  
    FOR jsmith
```



CREATE SYNCHRONIZATION SUBSCRIPTION

statement [MobiLink]

Description	Use this statement in an Adaptive Server Anywhere remote database to subscribe a MobiLink user to a publication.
Syntax	<pre> CREATE SYNCHRONIZATION SUBSCRIPTION TO <i>publication-name</i> [FOR <i>ml_username</i>, ...] [TYPE <i>sync-type</i>] [ADDRESS <i>network-parameters</i>] [OPTION <i>option=value</i>, ...] <i>ml_username</i>: identifier <i>network-parameters</i>: string <i>sync-type</i>: http https tcpip ActiveSync <i>value</i>: <i>string</i> <i>integer</i> </pre>
Parameters	<p>TO clause Specify the name of a publication.</p> <p>FOR clause Specify one or more MobiLink user names. <i>ml_username</i> is a name identifying a remote database. This name must be unique.</p> <p>☞ For more information about synchronization user names, see “About MobiLink users” [<i>MobiLink Synchronization User’s Guide</i>, page 104].</p> <p>Omit the FOR clause to set extended options, sync type and network parameters for a publication.</p> <p>☞ For information about how dbmlsync processes options that are specified in different locations, see “Priority order for extended options and connection parameters” [<i>MobiLink Synchronization User’s Guide</i>, page 180].</p> <p>TYPE clause This clause specifies the communication protocol to use for synchronization. The default protocol is tcpip.</p> <p>ADDRESS clause This clause specifies network parameters, including the location of the MobiLink synchronization server.</p> <p>☞ For a complete list of network parameters, see “CREATE SYNCHRONIZATION USER statement [MobiLink]” on page 245.</p> <p>OPTION clause This clause allows you to set extended options for the subscription. If no FOR clause is provided, the extended options act as default settings for the publication, and are overridden by any extended options set for a synchronization user.</p>


	☞ For a complete list of options, see “-e extended options” on page 44 .
Usage	Use this statement to create a synchronization subscription within a MobiLink remote or reference database.
Permissions	Must have DBA authority. Requires exclusive access to all tables referred to in the publication.
Side effects	Automatic commit.
See also	“CREATE PUBLICATION statement” on page 240 “CREATE SYNCHRONIZATION USER statement [MobiLink]” on page 245
Standards and compatibility	<p>◆ SQL/92 Vendor extension.</p> <p>◆ SQL/99 Vendor extension.</p>
Examples	<p>Create a default subscription, which contains default subscription values, for the sales publication (by omitting the FOR clause). Indicate the address of the MobiLink synchronization server and specify that only the Certicom root certificate is to be trusted.</p> <pre>CREATE SYNCHRONIZATION SUBSCRIPTION TO sales_publication ADDRESS 'host=test.internal;port=2439; security=ecc_tls' OPTION memory='2m';</pre> <p>Subscribe MobiLink user ml_user1 to the sales publication. Set the memory option to 3 Mb, rather than the value specified in the default publication.</p> <pre>CREATE SYNCHRONIZATION SUBSCRIPTION TO sales_publication FOR ml_user1 OPTION memory='3m';</pre>

CREATE SYNCHRONIZATION USER statement [MobiLink]

Description	Use this statement in an Adaptive Server Anywhere remote database to create a synchronization user.
Syntax	<pre> CREATE SYNCHRONIZATION USER <i>ml_username</i> [TYPE <i>sync-type</i>] [ADDRESS <i>network-parameters</i>] [OPTION <i>option=value, ...</i>] <i>ml_username</i>: identifier <i>sync-type</i>: tcip http https ActiveSync <i>network-parameters</i>: string <i>value</i>: string integer </pre>
Parameters	<p>ml_username A name identifying a remote database. This name must be unique.</p> <p> For more information about synchronization user names, see “About MobiLink users” [<i>MobiLink Synchronization User’s Guide</i>, page 104].</p> <p>TYPE clause This clause specifies the communication protocol to use for synchronization. The options are tcip, http, https, and ActiveSync. The default protocol is tcip.</p> <p>ADDRESS clause This clause specifies <i>network-parameters</i> in the form <i>keyword=value</i>, separated by semi-colons. Which settings you supply depends on the communication protocol you are using (TCP/IP, HTTP, HTTPS, or ActiveSync).</p> <p>♦ TCP/IP parameters If you specify the tcip protocol, you can optionally specify the following <i>network-parameters</i>:</p> <ul style="list-style-type: none"> • client_port=nnnnn or client_port=nnnnn-mmmmm A range of client ports for communication. If only one value is specified, the end of the range is 100 greater than the initial value, for a total of 101 ports. The option can be useful for clients inside a firewall communicating with a MobiLink synchronization server outside the firewall • host=hostname The host name or IP number for the machine on which the MobiLink synchronization server is running. The default value is localhost. For Windows CE, the default value is the value of <i>ipaddr</i> in the registry folder <i>Comm\Tcpip\Hosts\ppp_peer</i>. This allows a Windows CE device to connect to a MobiLink synchronization server

executing on the desktop machine where the Windows CE device's cradle is connected.

For the Palm Computing Platform, the default value of localhost refers to the device. It is recommended that an explicit host name or IP address be specified.

- **liveness_timeout=n** The amount of time, in seconds, after a client stops communicating before MobiLink recovers the connection. A value of 0 means that there is no timeout. This option is only effective if download acknowledgement on the client is set to off (the default). The default is 120 seconds.
- **port=portnumber** The socket port number. The port number must be a decimal number that matches the port the MobiLink synchronization server is setup to monitor. The default is 2439, which is the IANA registered port number for the MobiLink synchronization server.
- **network_name=name** Specify the network name so that you can use MobiLink's auto-dial feature. This allows you to connect from a Pocket PC 2002 or Windows desktop computer without manually dialing. Used with scheduling, your remote can synchronize unattended. Used without scheduling, this allows you to run dbmlsync without manually dialing a connection. The name should be the network name that you have specified in the dropdown list in Settings ► Connections ► Connections (Pocket PC) or Network & Dialup Connections (Windows).
 For more information about scheduling, see "Scheduling synchronization" [*MobiLink Synchronization User's Guide*, page 198].
- **network_connect_timeout=seconds** When you specify network_name, you can optionally specify a timeout after which the dial-up fails. This feature applies to Pocket PC 2002 only. (On Windows, you control this feature by configuring the connection profile.) The default is 120 seconds.
- **network_leave_open={0|1}** When you specify network_name, you can optionally specify that the connection should be left open after the synchronization finishes (1). The default behavior is to close the connection (0).
- **security=cipher(keyword=value;...)** All communication through this connection is to be encrypted using the cipher suite specified. The cipher can be one of **ecc_tls** or **rsa_tls**. These refer to elliptic-curve and RSA certification. For backwards compatibility, **ecc_tls** can also be specified as **certicom_tls**.

Separately licensable option required

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 about security, see “Transport-Layer Security” [*MobiLink Synchronization User’s Guide*, page 337].

The following security keywords are supported.

- **certificate_company** If you specify this parameter, the MobiLink client only accepts server certificates when the organization field on the certificate matches this value.
- **certificate_unit** If you specify this parameter, the MobiLink client only accepts server certificates when the organization unit field on the certificate matches this value.
- **certificate_name** If you specify this parameter, the MobiLink client only accepts server certificates when the common name field on the certificate matches this value.
- **trusted_certificates** When synchronization occurs through a Certicom TLS synchronization stream, the MobiLink synchronization server sends its certificate to the client, as well as the certificate of the entity that signed it, and so on up to a self-signed root.

The client checks that the chain is valid and that it trusts the root certificate in the chain. This feature allows you to specify which root certificates to trust. By default, the Sybase certificate is the trusted root.

- ◆ **HTTP parameters** If you specify the http protocol, you can optionally specify the following *network-parameters*:
 - **buffer_size=number** The maximum body size for a fixed content length message, in bytes. Changing the option will decrease or increase the amount of memory allocated for sending content. The default is 65 535, except on UltraLite and Pocket PC, in which case it is 1 024.
 - **client_port=nnnnn or client_port=nnnnn-mmmmm** A range of client ports for communication. If only one value is specified, the end of the range is 100 greater than the initial value, for a total of 101 ports. The option can be useful for clients inside a firewall communicating with a MobiLink synchronization server outside the firewall.
 - **host=hostname** The host name or IP number for the machine on which the MobiLink synchronization server is running. The default value is **localhost**. For Windows CE, the default value is the value of *ipaddr* in the registry folder *Comm\Tcpip\Hosts\ppp_peer*. This allows

a Windows CE device to connect to a MobiLink synchronization server executing on the desktop machine where the Windows CE device's cradle is connected.

For the Palm Computing Platform, the default value of `localhost` refers to the device. It is recommended that an explicit host name or IP address be specified.

- **network_name=name** Specify the network name so that you can use MobiLink's auto-dial feature. This allows you to connect from a Pocket PC 2002 or Windows desktop computer without manually dialing. Used with scheduling, your remote can synchronize unattended. Used without scheduling, this allows you to run `dbmlsync` without manually dialing a connection. The name should be the network name that you have specified in the dropdown list in Settings ► Connections ► Connections (Pocket PC) or Network & Dialup Connections (Windows).

☞ For more information about scheduling, see “Scheduling synchronization” [*MobiLink Synchronization User's Guide*, page 198].

- **network_connect_timeout=seconds** When you specify `network_name`, you can optionally specify a timeout after which the dial-up fails. This feature applies to Pocket PC 2002 only. (On Windows, you control this feature by configuring the connection profile.) The default is 120 seconds.
- **network_leave_open={0|1}** When you specify `network_name`, you can optionally specify that the connection should be left open after the synchronization finishes (1). The default behavior is to close the connection (0).
- **persistent={0|1}** 1 means that the client will attempt to use the same TCP/IP connection for all HTTP requests in a synchronization. A setting of 0 is more compatible with intermediate agents. The default is 1, except on Palm devices it is 0.

Note: Except on Palm devices, you should only set `persistent` to 1 if you are connecting directly to MobiLink. If you are connecting through an intermediate agent such as a proxy or redirector, a persistent connection may cause problems.

- **port=portnumber** The socket port number. The port number must be a decimal number that matches the port the MobiLink synchronization server is set up to monitor. The default value for the port number is **80**, which is the IANA registered port number for the MobiLink synchronization server.
- **proxy_host=proxy_hostname** The host name or IP address of the proxy server. The default value is **localhost**.

- **proxy_port=proxy_portnumber** The port number of the proxy server. The default value is **80**.
- **security=cipher(keyword=value;...)** All communication through this connection is to be encrypted using the cipher suite specified. The cipher can be one of **ecc_tls** or **rsa_tls**. These refer to elliptic-curve and RSA certification. For backwards compatibility, **ecc_tls** can also be specified as **certicom_tls**.

Separately licensable option required

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 about security, see “Transport-Layer Security” [*MobiLink Synchronization User’s Guide*, page 337].

The following security keywords are supported.

- **certificate_company** If you specify this parameter, the MobiLink client only accepts server certificates when the organization field on the certificate matches this value.
 - **certificate_unit** If you specify this parameter, the MobiLink client only accepts server certificates when the organization unit field on the certificate matches this value.
 - **certificate_name** If you specify this parameter, the MobiLink client only accepts server certificates when the common name field on the certificate matches this value.
 - **trusted_certificates** When synchronization occurs through a Certicom TLS synchronization stream, the MobiLink synchronization server sends its certificate to the client, as well as the certificate of the entity that signed it, and so on up to a self-signed root.
The client checks that the chain is valid and that it trusts the root certificate in the chain. This feature allows you to specify which root certificates to trust. By default, the Sybase certificate is the trusted root.
 - **url_suffix=suffix** The suffix to add to the URL on the first line of each HTTP request. When synchronizing through a proxy server, the suffix may be necessary in order to find the MobiLink synchronization server. The default value is **MobiLink**.
 - **version=versionnumber** A string specifying the version of HTTP to use. You have a choice of **1.0** or **1.1**. The default value is **1.1**.
- ♦ **HTTPS parameters** The HTTPS communication stream uses Certicom RSA security.

Separately licensable option required

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 about security, see “Transport-Layer Security” [*MobiLink Synchronization User’s Guide*, page 337].

If you specify the HTTPS protocol, you can optionally specify the following *network-parameters*:

- **buffer_size=number** The maximum body size for a fixed content length message, in bytes. Changing the option will decrease or increase the amount of memory allocated for sending content. The default is 65 535, except on UltraLite and Pocket PC, in which case it is 1 024.
- **client_port=nnnnn or client_port=nnnnn-mmmmm** A range of client ports for communication. If only one value is specified, the end of the range is 100 greater than the initial value, for a total of 101 ports. The option can be useful for clients inside a firewall communicating with a MobiLink synchronization server outside the firewall.
- **host=hostname** The host name or IP number for the machine on which the MobiLink synchronization server is running. The default value is **localhost**. For Windows CE, the default value is the value of *ipaddr* in the registry folder *Comm\Tcpip\Hosts\ppp_peer*. This allows a Windows CE device to connect to a MobiLink synchronization server executing on the desktop machine where the Windows CE device’s cradle is connected.

For the Palm Computing Platform, the default value of localhost refers to the device. It is recommended that an explicit host name or IP address be specified.

- **network_name=name** Specify the network name so that you can use MobiLink’s auto-dial feature. This allows you to connect from a Pocket PC 2002 or Windows desktop computer without manually dialing. Used with scheduling, your remote can synchronize unattended. Used without scheduling, this allows you to run dbmlsync without manually dialing a connection. The name should be the network name that you have specified in the dropdown list in Settings ► Connections ► Connections (Pocket PC) or Network & Dialup Connections (Windows).

☞ For more information about scheduling, see “Scheduling synchronization” [*MobiLink Synchronization User’s Guide*, page 198].

- **network_connect_timeout=seconds** When you specify *network_name*, you can optionally specify a timeout after which the dial-up fails. This feature applies to Pocket PC 2002 only. (On

Windows, you control this feature by configuring the connection profile.) The default is 120 seconds.

- **network_leave_open={0|1}** When you specify `network_name`, you can optionally specify that the connection should be left open after the synchronization finishes (1). The default behavior is to close the connection (0).
- **persistent={0|1}** 1 means that the client will attempt to use the same TCP/IP connection for all HTTP requests in a synchronization. A setting of 0 is more compatible with intermediate agents. The default is 1, except on Palm devices it is 0.

Note: Except on Palm devices, you should only set `persistent` to 1 if you are connecting directly to MobiLink. If you are connecting through an intermediate agent such as a proxy or redirector, a persistent connection may cause problems.

- **port=portnumber** The socket port number. The port number must be a decimal number that matches the port the MobiLink synchronization server is set up to monitor. The default value for the port parameter is **443**, which is the IANA registered port number for the MobiLink synchronization server.
- **proxy_host=proxy_hostname** The host name or IP address of the proxy server. The default value is **localhost**.
- **proxy_port=proxy_portnumber** The port number of the proxy server. The default value is **443**.
- **certificate_company** If you specify this parameter, the MobiLink client only accepts server certificates when the organization field on the certificate matches this value.
- **certificate_unit** If you specify this parameter, the MobiLink client only accepts server certificates when the organization unit field on the certificate matches this value.
- **certificate_name** If you specify this parameter, the MobiLink client only accepts server certificates when the common name field on the certificate matches this value.
- **trusted_certificates** When synchronization occurs through a Certicom TLS synchronization stream, the MobiLink synchronization server sends its certificate to the client, as well as the certificate of the entity that signed it, and so on up to a self-signed root.

The client checks that the chain is valid and that it trusts the root certificate in the chain. This feature allows you to specify which root certificates to trust. By default, the Sybase certificate is the trusted root.

☞ For more information about security, see “Transport-Layer Security” [*MobiLink Synchronization User’s Guide*, page 337].


- **url_suffix=suffix** The suffix to add to the URL on the first line of each HTTPS request. When synchronizing through a proxy server, the suffix may be necessary in order to find the MobiLink synchronization server. The default value is **MobiLink**.
- **version=versionnumber** A string specifying the version of HTTP to use. You have a choice of **1.0** or **1.1**. The default value is **1.1**.

- ◆ **ActiveSync parameters** During ActiveSync synchronization, ActiveSync is used to exchange data with the MobiLink provider for ActiveSync, which resides on the desktop machine. The ActiveSync parameters describe the communications between the MobiLink provider for ActiveSync and the MobiLink synchronization server.

The address string for ActiveSync takes the following form:

stream=desktop-stream;[desktop-stream-params]

where:

- *desktop-stream* is the synchronization stream to use between the MobiLink provider for ActiveSync and the MobiLink synchronization server. It can be **http**, **https**, or **tcpip**. The default setting is **tcpip**.
- *desktop-stream-params* are TCP/IP, HTTP, or HTTPS parameters, as described in the lists above.
 For more information, see [“ActiveSync provider installation utility” on page 300](#).

OPTION clause The OPTION clause allows you to set options using *option=value* in a comma-separated list.

The values for each option cannot contain equal signs or semicolons. The database server accepts any option that you enter without checking for its validity. Therefore, if you misspell an option or enter an invalid value, no error message appears until you run the dbmlsync command to perform synchronization.

Options set for a synchronization user can be overridden in individual subscriptions or on the dbmlsync command line.

For complete information about extended options, see [“-e extended options” on page 44](#).

Description

The *sync-type*, *network-parameters*, and *options* can be set in several places:

- ◆ on the dbmlsync command line using the -e or -eu options
- ◆ in Sybase Central
- ◆ using the following SQL statements:

- CREATE SYNCHRONIZATION SUBSCRIPTION
- ALTER SYNCHRONIZATION SUBSCRIPTION
- CREATE SYNCHRONIZATION USER
- ALTER SYNCHRONIZATION USER
- CREATE SYNCHRONIZATION SUBSCRIPTION without specifying a synchronization user (this associates the values with a publication)

When you store extended options and connection parameters in the database, dbmlsync reads the information from the database. If values are specified in both the database and the command line, the value strings are combined. If conflicting values are specified, dbmlsync resolves them as follows, where values occurring earlier in the list take precedence over those occurring later in the list.

- ◆ dbmlsync extended option -eu
- ◆ dbmlsync extended option -e
- ◆ specified on the subscription (whether by a SQL statement or in Sybase Central)
- ◆ specified on the MobiLink user (whether by a SQL statement or in Sybase Central)
- ◆ specified on the publication (whether by a SQL statement or in Sybase Central)

Permissions

Must have DBA authority.

Side effects

Automatic commit.

See also

“ALTER SYNCHRONIZATION USER statement [MobiLink]” on page 238
 “CREATE SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]” on page 243
 “CREATE PUBLICATION statement” on page 240
 “-e extended options” on page 44

Standards and compatibility

- ◆ **SQL/92** Vendor extension.
- ◆ **SQL/99** Vendor extension.
- ◆ **Sybase** Supported by Open Client/Open Server.

Examples

The following example creates a user named SSinger, who synchronizes over TCP/IP with a server machine named mlserver.mycompany.com using the password Sam. The use of a password in the user definition is *not* secure.

```
CREATE SYNCHRONIZATION USER SSinger
TYPE http
ADDRESS 'host=mlserver.mycompany.com'
OPTION MobiLinkPwd='Sam'
```

The following creates a synchronization user called factory014 that will cause dbmlsync to hover and synchronize via Certicom-encrypted TCP/IP at a random time in every 8-hour interval. The randomness helps prevent performance degradation at the MobiLink server due to multiple simultaneous synchronizations:

```
CREATE SYNCHRONIZATION USER factory014
TYPE tcpip
ADDRESS 'host=mycompany.manufacturing.mobilink1;security=certico
        m_tls(certificate=mycompany_mobilink.crt;certificate_
        password=thepassword)'
OPTION Schedule='EVERY:08:00'
```

The following creates a synchronization user called sales5322 that will be used to synchronize with HTTP. In this example, the MobiLink synchronization server runs behind the corporate firewall, and synchronization requests are redirected to it using the Redirector (a reverse proxy to an NSAPI Web server).

```
CREATE SYNCHRONIZATION USER sales5322
TYPE https
ADDRESS 'host=www.mycompany.com;port=80;url_
        suffix=mlredirect/ml/'
```

DROP PUBLICATION statement

Description	Use this statement to drop a publication. In MobiLink a publication identifies synchronized data in a Adaptive Server Anywhere remote database. In SQL Remote, publications identify replicated data in both consolidated and remote databases.
Syntax	DROP PUBLICATION [<i>owner</i> .] <i>publication-name</i> <i>owner</i> , <i>publication-name</i> : <i>identifier</i>
Usage	This statement is applicable only to MobiLink and SQL Remote.
Permissions	Must have DBA authority.
Side effects	Automatic commit. All subscriptions to the publication are dropped.
See also	“ALTER PUBLICATION statement” on page 234 “CREATE PUBLICATION statement” on page 240 “sp_drop_publication procedure” [SQL Remote User’s Guide, page 385]
Standards and compatibility	♦ SQL/92 Vendor extension. ♦ SQL/99 Vendor extension.
Example	The following statement drops the pub_contact publication. <pre>DROP PUBLICATION pub_contact</pre>

DROP SYNCHRONIZATION SUBSCRIPTION

statement [MobiLink]

Description	Use this statement to drop a synchronization subscription within a MobiLink remote database or a MobiLink reference database. You can also use it to drop a default subscription, which contains default subscription values, for the specified publication.
Syntax	DROP SYNCHRONIZATION SUBSCRIPTION TO <i>publication-name</i> [FOR <i>ml_username</i> , ...]
Parameters	TO clause Specify the name of a publication. FOR clause Specify one more MobiLink users. Omitting this clause drops the default subscription for the publication. MobiLink users subscribed to a publication inherit as defaults the values in a default publication.
Usage	Drop a synchronization subscription in a MobiLink remote or reference database.
Permissions	Must have DBA authority. Requires exclusive access to all tables referred to in the publication.
Side Effects	Automatic commit.
Standards and compatibility	♦ SQL/92 Vendor extension. ♦ SQL/99 Vendor extension.
Examples	Unsubscribe MobiLink user ml_user1 to the sales publication. <pre>DROP SYNCHRONIZATION SUBSCRIPTION TO sales_publication FOR "ml_user1"</pre> Drop the default subscription, which contains default subscription values, for the sales publication (by omitting the FOR clause). <pre>DROP SYNCHRONIZATION SUBSCRIPTION TO sales_publication</pre>

DROP SYNCHRONIZATION USER statement [MobiLink]

Description	Use this statement to drop a synchronization user from a MobiLink remote database.
Syntax	DROP SYNCHRONIZATION USER <i>ml_username</i> , ... <i>ml_username</i> : <i>identifier</i>
Usage	Drop one or more synchronization users from a MobiLink remote database.
Permissions	Must have DBA authority. Requires exclusive access to all tables referred to in the publication.
Side Effects	All subscriptions associated with the user are also deleted.
Standards and compatibility	♦ SQL/92 Vendor extension. ♦ SQL/99 Vendor extension.
Example	Remove MobiLink user ml_user1 from the database. <pre>DROP SYNCHRONIZATION USER ml_user1</pre>

START SYNCHRONIZATION DELETE statement [MobiLink]

Description	Use this statement to restart logging of deletes for MobiLink synchronization.
Syntax	START SYNCHRONIZATION DELETE
Usage	<p>Ordinarily, Adaptive Server Anywhere automatically logs any changes made to tables or columns that are part of a synchronization template and uploads these changes to the consolidated database during the next synchronization. You can temporarily suspend automatic logging of delete operations using the STOP SYNCHRONIZATION DELETE statement. The START SYNCHRONIZATION DELETE statement allows you to restart the automatic logging.</p> <p>When a STOP SYNCHRONIZATION DELETE statement is executed, none of the delete operations executed on that connection will be synchronized. The effect continues until a START SYNCHRONIZATION DELETE statement is executed. The effects do not nest; that is, subsequent execution of stop synchronization delete after the first will have no additional effect. A single START SYNCHRONIZATION DELETE statement restarts the logging, regardless of the number of STOP SYNCHRONIZATION DELETE statements preceding it.</p>
Permissions	Must have DBA authority.
Side effects	None.
See also	<p>“STOP SYNCHRONIZATION DELETE statement [MobiLink]” on page 260</p> <p>“StartSynchronizationDelete method” [<i>UltraLite Static C++ User’s Guide</i>, page 86]</p>
Standards and compatibility	<ul style="list-style-type: none">◆ SQL/92 Vendor extension.◆ SQL/99 Vendor extension.◆ Sybase Not applicable.
Example	The following sequence of SQL statements illustrates how to use START SYNCHRONIZATION DELETE and STOP SYNCHRONIZATION DELETE.



```
-- Prevent deletes from being sent
-- to the consolidated database
STOP SYNCHRONIZATION DELETE;

-- Remove all records older than 1 month
-- from the remote database,
-- NOT the consolidated database
DELETE FROM PROPOSAL
WHERE last_modified < months( CURRENT_TIMESTAMP, -1 )

-- Re-enable all deletes to be sent
-- to the consolidated database
-- DO NOT FORGET to start this
START SYNCHRONIZATION DELETE;

-- Commit the entire operation,
-- otherwise rollback everything
-- including the stopping of the deletes
commit;
```

STOP SYNCHRONIZATION DELETE statement [MobiLink]

Description	Use this statement to temporarily stop logging of deletes for MobiLink synchronization.
Syntax	STOP SYNCHRONIZATION DELETE
Usage	<p>Ordinarily, Adaptive Server Anywhere automatically logs any changes made to tables or columns that are part of a synchronization template and uploads these changes to the consolidated database during the next synchronization. This statement allows you to temporarily suspend logging of changes to an Adaptive Server Anywhere remote database.</p> <p>When a STOP SYNCHRONIZATION DELETE statement is executed, none of the subsequent delete operations executed on that connection will be synchronized. The effect continues until a START SYNCHRONIZATION DELETE statement is executed. The effects do not nest; that is, subsequent execution of stop synchronization delete after the first will have no additional effect. A single START SYNCHRONIZATION DELETE statement restarts the logging, regardless of the number of STOP SYNCHRONIZATION DELETE statements preceding it.</p> <p>This command can be useful to make corrections to a remote database, but should be used with caution as it effectively disables MobiLink synchronization.</p>
Permissions	Must have DBA authority.
Side Effects	None.
See also	<p>“StartSynchronizationDelete method” [<i>UltraLite Static C++ User’s Guide</i>, page 86]</p> <p>“StopSynchronizationDelete method” [<i>UltraLite Static C++ User’s Guide</i>, page 86]</p>
Standards and compatibility	<ul style="list-style-type: none">◆ SQL/92 Vendor extension.◆ SQL/99 Vendor extension.◆ Sybase Not applicable.
Example	 For an example, see “START SYNCHRONIZATION DELETE statement [MobiLink]” on page 258.

CHAPTER 5

Stored Procedures

About this chapter

This chapter provides information about the MobiLink pre-defined stored procedures. There are two types of stored procedure:

- ◆ **Server stored procedures** facilitate management of synchronization scripts using SQL statements. They are stored in the consolidated database.
- ◆ **Client stored procedures** perform specific tasks during synchronization on an Adaptive Server Anywhere client. These are called client event-hook procedures.

Contents

Topic:	page
Stored procedures to add or delete scripts	262
Client event-hook procedures	269

Stored procedures to add or delete scripts

You must add synchronization scripts to system tables in the consolidated database before you can use them. The following stored procedures add synchronization scripts to the consolidated database. They can also be used to delete scripts.

- Notes
- ◆ When you add a script using a stored procedure, the script is a string. Any strings within the script need to be escaped. For Adaptive Server Anywhere, each quotation mark (‘) needs to be doubled so as not to terminate the string.
 - ◆ You cannot use stored procedures to add scripts longer than 255 bytes to Adaptive Server Enterprise 11.5 or earlier. Instead, use Sybase Central or direct insertion to define longer scripts.
 - ◆ IBM DB2 prior to version 6 only supports column names and other identifiers of 18 characters or less, and so the names are truncated. For example, ml_add_connection_script is shortened to ml_add_connection_.

ml_add_connection_script

Function

Use this stored procedure to add or delete SQL connection scripts in the consolidated database.

Parameters

Item	Parameter	Description
1	version	CHAR(128)
2	event	CHAR(128)
3	script	For Adaptive Server Anywhere and MS SQL Server, this is TEXT. For ASE, this is VARCHAR(16384). For ASE prior to 12.5, this is VARCHAR(255). For DB2, this is VARCHAR(4000). For Oracle, this is VARCHAR.

Description

To delete a connection script, set the script parameter to NULL.

When you add a script, the script is inserted into the ml_script table and the appropriate references are defined to associate the script with the event and script version that you specify. If the version name is new, it is automatically inserted into the ml_version table.

See also

“Adding and deleting scripts in your consolidated database” [*MobiLink Synchronization User’s Guide*, page 51]

[“ml_add_table_script” on page 263](#)

[“ml_add_dnet_connection_script” on page 264](#)

[“ml_add_dnet_table_script” on page 265](#)

[“ml_add_java_connection_script” on page 266](#)

[“ml_add_java_table_script” on page 267](#)

Example

The following statement adds a connection script associated with the `begin_synchronization` event to the script version `custdb` in an Adaptive Server Anywhere consolidated database. The script itself is the single statement that sets the `@EmployeeID` variable.

```
call ml_add_connection_script( 'custdb',
    'begin_synchronization',
    'set @EmployeeID = ?' )
```

ml_add_table_script

Function

Use this stored procedure to add or delete SQL table scripts in the consolidated database.

Parameters

Item	Parameter	Description
1	version	VARCHAR(128)
2	table_name	VARCHAR(128)
3	event	VARCHAR(128)
4	script	For Adaptive Server Anywhere and MS SQL Server, this is TEXT. For ASE, this is VARCHAR(16384). For ASE prior to 12.5, this is VARCHAR(255). For DB2, this is VARCHAR(4000). For Oracle, this is VARCHAR.

Description

To delete a table script, set the script parameter to NULL.

When you add a script, the script is inserted into the `ml_script` table and the appropriate references are defined to associate the script with the table, event and script version that you specify. If the version name is new, it is automatically inserted into the `ml_version` table.

See also

“Adding and deleting scripts in your consolidated database” [*MobiLink Synchronization User’s Guide*, page 51]

[“ml_add_connection_script” on page 262](#)

[“ml_add_dnet_connection_script” on page 264](#)

[“ml_add_dnet_table_script” on page 265](#)

[“ml_add_java_connection_script” on page 266](#)

[“ml_add_java_table_script” on page 267](#)

Example

The following command adds a cursor script associated with the upload_cursor event on the ULCustomer table.

```
call ml_add_table_script( 'custdb',
    'ULCustomer',
    'upload_cursor',
    'SELECT cust_id, cust_name
    FROM ULCustomer WHERE cust_id = ?' )
```

ml_add_dnet_connection_script

Function

Use this stored procedure to add or delete .NET connection scripts in the consolidated database.

Parameters

Item	Parameter	Description
1	version	CHAR(128)
2	event	CHAR(128)
3	script	For Adaptive Server Anywhere and MS SQL Server, this is TEXT. For ASE, this is VARCHAR(16384). For ASE prior to 12.5, this is VARCHAR(255). For DB2, this is VARCHAR(4000). For Oracle, this is VARCHAR.

Description

To delete a connection script, set the script parameter to NULL.

The *script* value is a public method in a class in the MobiLink synchronization server classpath (for example, MyClass.MyMethod).

When you add a script, the method is associated with the event and script version that you specify. If the version name is new, it is automatically inserted into the ml_version table.

See also

“Adding and deleting scripts in your consolidated database” [*MobiLink Synchronization User’s Guide*, page 51]

[“ml_add_dnet_table_script” on page 265](#)

[“ml_add_connection_script” on page 262](#)

[“ml_add_table_script” on page 263](#)

[“ml_add_java_table_script” on page 267](#)

“Methods” [*MobiLink Synchronization User’s Guide*, page 234]

Example

The following example assigns the beginDownloadConnection method of the ExampleClass class to the begin_download event.

```
call ml_add_dnet_connection_script( 'ver1',
'begin_download',
'ExamplePackage.ExampleClass.beginDownloadConnection')
```

ml_add_dnet_table_script

Function

Use this stored procedure to add or delete .NET table scripts in the consolidated database.

Parameters

Item	Parameter	Description
1	version	VARCHAR(128)
2	table	VARCHAR(128)
3	event	VARCHAR(128)
4	script	For Adaptive Server Anywhere and MS SQL Server, this is TEXT. For ASE, this is VARCHAR(16384). For ASE prior to 12.5, this is VARCHAR(255). For DB2, this is VARCHAR(4000). For Oracle, this is VARCHAR.

Description

To delete a connection script, set the script parameter to NULL.

The *script* value is a public method in a class in the MobiLink synchronization server classpath (for example, MyClass.MyMethod).

When you add a script, the method is associated with the table, event, and script version that you specify. If the version name is new, it is automatically inserted into the ml_version table.

See also

“Adding and deleting scripts in your consolidated database” [*MobiLink Synchronization User’s Guide*, page 51]

[“ml_add_dnet_connection_script” on page 264](#)

[“ml_add_connection_script” on page 262](#)

[“ml_add_table_script” on page 263](#)

[“ml_add_java_connection_script” on page 266](#)

“Methods” [*MobiLink Synchronization User’s Guide*, page 234]

Example

The following example assigns the empDownloadCursor method of the EgClass class to the download_cursor event for the table emp.

```
call ml_add_dnet_table_script( 'ver1', 'emp',  
    'download_cursor', EgPackage.EgClass.empDownloadCursor )
```

ml_add_java_connection_script

Function

Use this stored procedure to add or delete Java connection scripts in the consolidated database.

Parameters

Item	Parameter	Description
1	version	CHAR(128)
2	event	CHAR(128)
3	script	For Adaptive Server Anywhere and MS SQL Server, this is TEXT. For ASE, this is VARCHAR(16384). For ASE prior to 12.5, this is VARCHAR(255). For DB2, this is VARCHAR(4000). For Oracle, this is VARCHAR.

Description

To delete a connection script, set the script parameter to NULL.

The *script* value is a public method in a class in the MobiLink synchronization server classpath (for example, MyClass.MyMethod).

When you add a script, the method is associated with the event and script version that you specify. If the version name is new, it is automatically inserted into the ml_version table.

See also

“Adding and deleting scripts in your consolidated database” [*MobiLink Synchronization User’s Guide*, page 51]

[“ml_add_connection_script” on page 262](#)

[“ml_add_table_script” on page 263](#)

[“ml_add_dnet_connection_script” on page 264](#)

[“ml_add_dnet_table_script” on page 265](#)

[“ml_add_java_table_script” on page 267](#)

“Methods” [*MobiLink Synchronization User’s Guide*, page 234]

Example

The following example is taken from the *Samples\MobiLink\JavaAuthentication* sample. It assigns the `endConnection` method of the `CustEmpScripts` class to the `end_connection` event.

```
call ml_add_java_connection_script( 'ver1',
    'end_connection',
    'CustEmpScripts.endConnection')
```

ml_add_java_table_script

Function

Use this stored procedure to add or delete Java table scripts in the consolidated database.

Parameters

Item	Parameter	Description
1	version	VARCHAR(128)
2	table	VARCHAR(128)
3	event	VARCHAR(128)
4	script	For Adaptive Server Anywhere and MS SQL Server, this is TEXT. For ASE, this is VARCHAR(16384). For ASE prior to 12.5, this is VARCHAR(255). For DB2, this is VARCHAR(4000). For Oracle, this is VARCHAR.

Description

To delete a connection script, set the script parameter to NULL.

The *script* value is a public method in a class in the MobiLink synchronization server classpath (for example, `MyClass.MyMethod`).

When you add a script, the method is associated with the table, event, and script version that you specify. If the version name is new, it is automatically inserted into the `ml_version` table.

See also

“Adding and deleting scripts in your consolidated database” [*MobiLink Synchronization User’s Guide*, page 51]

[“ml_add_connection_script” on page 262](#)

[“ml_add_table_script” on page 263](#)

[“ml_add_dnet_connection_script” on page 264](#)

[“ml_add_dnet_table_script” on page 265](#)

[“ml_add_java_connection_script” on page 266](#)

“Methods” [*MobiLink Synchronization User’s Guide*, page 234]

Example

The following example is taken from the *Samples\MobiLink\JavaAuthentication* sample. It assigns the `empDownloadCursor` method of the `CustEmpScripts` class to the `download_cursor` event for the table `emp`.

```
call ml_add_java_table_script( 'ver1', 'emp',  
    'download_cursor','CustEmpScripts.empDownloadCursor')
```

Client event-hook procedures

The following stored procedures provide the interface for customizing synchronization at Adaptive Server Anywhere clients.

Notes

- ◆ Do not perform any COMMIT or ROLLBACK operations in event-hook procedures. The procedures are executed on the same connection as the synchronization, and a COMMIT or ROLLBACK may interfere with synchronization.
- ◆ Do not define more than one hook with the same name. If more than one hook with the same name is created (say by different users), then which hook is called is undefined.
- ◆ Hook procedures must be created by a user with DBA authority.
- ◆ If a *_begin hook executes successfully, the corresponding *_end hook is called regardless of any error that occurs afterwards. If the *_begin hook is not defined, but you have defined an *_end hook, then the *_end hook is called unless an error occurs prior the point in time where the *_begin hook would normally be called.

☞ For more information about using client event hooks, see “Customizing the client synchronization process” [*MobiLink Synchronization User’s Guide*, page 194].

The #hook_dict table

Immediately before a hook is called, dbmlsync creates the #hook_dict table in the remote database, using the following CREATE statement. The # before the table name means that the table is temporary.

```
CREATE TABLE #hook_dict(
  name VARCHAR(128) NOT NULL UNIQUE,
  value VARCHAR(255) NOT NULL)
```

dbmlsync uses the #hook_dict table to pass values to hook functions, and hook functions use the #hook_dict table to pass values back to dbmlsync.

For example, for the following dbmlsync command line,

```
dbmlsync -c 'dsn=MyDsn' -n pub1, pub2 -u MyUser
```

when the sp_hook_dbmlsync_abort hook is called, the #hook_dict table will contain the following rows:

Name	Value
publication_0	pub1

Name	Value
publication_1	pub2
MobiLink user	MyUser
Abort synchronization	false

Your abort hook can retrieve values from the #hook_dict table and use them to customize behavior. For example, to retrieve the MobiLink user you would use a SELECT statement like this:

```
SELECT value
FROM #hook_dict
WHERE name = 'MobiLink user'
```

In/out parameters can be updated by your hook to modify the behavior of dbmlsync. For example, your hook could instruct dbmlsync to abort synchronization by updating the abort synchronization row of the table using a statement like this:

```
UPDATE #hook_dict
SET value='true'
WHERE name='abort synchronization'
```

The description of each hook lists the rows in the #hook_dict table.

sp_hook_dbmlsync_abort

Function

Use this stored procedure to cancel the synchronization process; or to log exit codes.

Rows in #hook_dict table

Name	Values	Description
abort synchronization (in/out)	True False	If you set the abort synchronization row of the #hook_dict table to true , then dbmlsync terminates immediately after the event.
publication_n (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one publication_ <i>n</i> entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
return code (in)	number	When abort synchronization is set to TRUE, you can use this value to set the return code for the aborted synchronization. 0 indicates a successful synchronization. Any other number indicates that the synchronization failed.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description If a procedure of this name exists, it is called at dbmlsync startup, and then again after each synchronization delay that is caused by the sp_hook_dbmlsync_delay hook.

If the hook requests an abort by setting the abort synchronization value to true, the exit code is passed to the sp_hook_dbmlsync_process_exit_code hook. If no sp_hook_dbmlsync_process_exit_code hook is defined, the exit code is used as the exit code for the program.

Actions of this procedure are committed immediately after execution.

See also “Synchronization event hook sequence” [*MobiLink Synchronization User’s Guide*, page 194]

[“sp_hook_dbmlsync_process_return_code” on page 289](#)

Examples The following procedure prevents synchronization during a scheduled maintenance hour between 19:00 and 20:00 each day.

```

create procedure sp_hook_dbmlsync_abort()
begin
  declare down_time_start time;
  declare is_down_time varchar(128);
  set down_time_start='19:00';
  if abs( datediff( hour,down_time_start,now(*) ) ) < 1
  then
    set is_down_time='true';
  else
    set is_down_time='false';
  end if;
  UPDATE #hook_dict
  SET value = is_down_time
  WHERE name = 'abort synchronization'
end

```

Suppose you have an abort hook that may abort synchronization for one of two reasons. One of the reasons indicates normal completion of synchronization, so you want dbmlsync to have an exit code of 0. The other reason indicates an error condition, so you want dbmlsync to have a non-zero exit code. You could achieve this with an `sp_hook_dbmlsync_abort` hook defined as follows.

```

BEGIN
  IF [condition that defines the normal abort case] THEN
    UPDATE #hook_dict SET value = '0' WHERE name = 'return
      code';
    UPDATE #hook_dict SET value = 'TRUE' WHERE name = 'abort
      synchronization';
  ELSEIF [condition that defines the error abort case] THEN
    UPDATE #hook_dict SET value = '1' WHERE name = 'return
      code';
    UPDATE #hook_dict SET value = 'TRUE' WHERE name = 'abort
      synchronization';
  END IF;
END;

```

sp_hook_dbmlsync_begin

Function Use this stored procedure to add custom actions at the beginning of the synchronization process.

Rows in #hook_dict table

Name	Values	Description
publication_ <i>n</i> (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one publication_ <i>n</i> entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description If a procedure of this name exists, it is called at the beginning of the synchronization process.

Actions of this procedure are committed immediately after execution.

See also “Synchronization event hook sequence” [*MobiLink Synchronization User’s Guide*, page 194]

sp_hook_dbmlsync_delay

Function Use this stored procedure to control when synchronization takes place.

Rows in #hook_dict table

Name	Values	Description
delay duration (in out)	number of seconds	If the procedure sets the delay duration value to zero, then dbmlsync synchronization proceeds. A non-zero delay_duration value specifies the number of seconds before the delay hook is called again.
maximum accumulated delay (in out)	number of seconds	The maximum accumulated delay specifies the maximum number of seconds delay before each synchronization. Dbmlsync keeps track of the total delay created by all calls to the delay hook since the last synchronization. If no synchronization has occurred since dbmlsync started running, the total delay is calculated from the time dbmlsync started up. When the total delay exceeds the value of maximum accumulated delay, synchronization begins without any further calls to the delay hook.
publication_n (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one publication_n entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description	<p>If a procedure of this name exists, it is called before sp_hook_dbmlsync_begin at the beginning of the synchronization process.</p> <p>Actions of this procedure are committed immediately after execution.</p>
See also	“Synchronization event hook sequence” [<i>MobiLink Synchronization User’s Guide</i> , page 194]
Example	The following procedure delays synchronization during a scheduled

maintenance hour between 19:00 and 20:00 each day.

```
create procedure sp_hook_dbmlsync_delay()
begin
  declare down_time_start time;
  declare is_down_time varchar(128);
  set down_time_start='19:00';
  if abs( datediff( minute,down_time_start,now(*) ) ) <
  60 then
    set is_down_time='10';
  else
    set is_down_time='0';
  end if;
  UPDATE #hook_dict
  SET value = is_down_time
  WHERE name = 'delay duration'
end
```

sp_hook_dbmlsync_download_begin

Function Use this stored procedure to add custom actions at the beginning of the download stage of the synchronization process.

Rows in #hook_dict table

Name	Values	Description
publication_n (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one publication_n entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description If a procedure of this name exists, it is called at the beginning of the download stage of the synchronization process.

Actions of this procedure are committed or rolled back when the download stream is committed or rolled back.

See also “Synchronization event hook sequence” [*MobiLink Synchronization User’s Guide*, page 194]

sp_hook_dbmlsync_download_com_error

Function Use this stored procedure to add custom actions when communications

errors occur while reading the download stream sent by the MobiLink synchronization server.

Rows in #hook_dict table

Name	Values	Description
table name (in)	table name	The table to which operations were being applied when the error occurred. The value is an empty string if dbmlsync is unable to identify the table.
publication_n (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one publication_n entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description

If a procedure of this name exists, it is invoked when a communication error is detected during the download phase of synchronization. The download is then terminated.

This procedure executes on a separate connection, so that failures can be logged. Otherwise, the action of logging would be rolled back along with the synchronization actions. If dbmlsync cannot establish a separate connection, the procedure is not called.

By default on Windows CE devices, synchronization tables are locked in exclusive mode, which means that this hook cannot successfully execute if it requires access to any of the synchronization tables. It also cannot execute if it needs to access synchronization tables and you set the dbmlsync extended option LockTables to EXCLUSIVE. For more information, see [“LockTables \(It\) extended option” on page 55](#).

Actions of this procedure are committed immediately after execution.

See also

“Synchronization event hook sequence” [*MobiLink Synchronization User’s Guide*, page 194]

sp_hook_dbmlsync_download_end

Function

Use this stored procedure to add custom actions at the end of the download stage of the synchronization process.

Rows in #hook_dict table

Name	Values	Description
publication_n (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one <i>publication_n</i> entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description If a procedure of this name exists, it is called at the end of the download stage of the synchronization process.

Actions of this procedure are committed or rolled back when the download stream is committed or rolled back.

See also “Synchronization event hook sequence” [*MobiLink Synchronization User’s Guide*, page 194]

sp_hook_dbmlsync_download_fatal_sql_error

Function Take action when a synchronization download is about to be rolled back because of a database error.

Rows in #hook_dict table

Name	Values	Description
table name (in)	table name	The table to which operations were being applied when the error occurred. The value is an empty string if dbmlsync is unable to identify the table.
SQL error code (in)	SQL error code	Identifies the SQL error code returned by the database when the operation failed.
publication_n (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one publication_ <i>n</i> entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description

If a procedure of this name exists, it is called immediately before a synchronization download is rolled back because of a database error. This occurs whenever an SQL error is encountered that cannot be ignored, or when the `sp_hook_dbmlsync_download_SQL_error` hook has already been called and has chosen not to ignore the error.

This procedure executes on a separate connection, so that failures can be logged. Otherwise, the action of logging would be rolled back along with the synchronization actions. If dbmlsync cannot establish a separate connection, the procedure is not called.

By default on Windows CE devices, synchronization tables are locked in exclusive mode, which means that this hook cannot successfully execute if it requires access to any of the synchronization tables. It also cannot execute if it needs to access synchronization tables and you set the dbmlsync extended option `LockTables` to `EXCLUSIVE`. For more information, see [“LockTables \(It\) extended option” on page 55](#).

Actions of this procedure are committed immediately after execution.

See also

“Synchronization event hook sequence” [*MobiLink Synchronization User’s Guide*, page 194]

[“sp_hook_dbmlsync_download_sql_error” on page 282](#)

sp_hook_dbmsync_download_log_ri_violation

Function Logs referential integrity violations in the download process.

Rows in #hook_dict table

Name	Values	Description
publication_n (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one <i>publication_n</i> entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
Foreign key table (in)	table name	The table containing the foreign key column for which the hook is being called.
Primary key table (in)	table name	The table referenced by the foreign key for which the hook is being called.
Role name (in)	role name	The role name of the foreign key for which the hook is being called.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description A download RI violation occurs when rows in the download stream violate foreign key relationships on the remote database. This hook allows you to log RI violations as they occur so that you can investigate their cause later.

After the download is complete, but before it is committed, dbmsync checks for RI violations. If it finds any, it identifies a foreign key that has an RI violation and calls `sp_hook_dbmsync_download_log_ri_violation` (if it is implemented). It then calls `sp_hook_dbmsync_download_ri_conflict` (if it is implemented). If there is still a conflict, dbmsync deletes the rows that violate the foreign key constraint. This process is repeated for remaining foreign keys that have RI violations.

This hook is called only when there are RI violations involving tables that are currently being synchronized. If there are RI violations involving tables that are not being synchronized, this hook is not called and the synchronization fails.

This hook is called on a separate connection from the one that dbmsync

uses for the download. The connection used by the hook has an isolation level of 0 so that the hook can see the rows that have been applied from the download stream that are not yet committed. The actions of the hook are committed immediately after it completes so that changes made by this hook will be preserved regardless of whether the download stream is committed or rolled back.

By default on Windows CE devices, synchronization tables are locked in exclusive mode, which means that this hook cannot successfully execute if it requires access to any of the synchronization tables. It also cannot execute if it needs to access synchronization tables and you set the dbmlsync extended option LockTables to EXCLUSIVE. For more information, see [“LockTables \(It\) extended option” on page 55](#).

Do not attempt to use this hook to correct RI violation problems. It should be used for logging only. Use `sp_hook_dbmlsync_download_ri_violation` to resolve RI violations.

See also [“sp_hook_dbmlsync_download_ri_violation” on page 280](#)

sp_hook_dbmlsync_download_ri_violation

Function	Allows you to resolve referential integrity violations in the download process.
----------	---

Rows in #hook_dict table

Name	Values	Description
publication_ <i>n</i> (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one publication_ <i>n</i> entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
Foreign key table (in)	table name	The table containing the foreign key column for which the hook is being called.
Primary key table (in)	table name	The table referenced by the foreign key for which the hook is being called.
Role name (in)	role name	The role name of the foreign key for which the hook is being called.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description

A download RI violation occurs when rows in the download stream violate foreign key relationships on the remote database. This hook allows you to attempt to resolve RI violations before dbmlsync deletes the rows that are causing the conflict.

After the download is complete, but before it is committed, dbmlsync checks for RI violations. If it finds any, it identifies a foreign key that has an RI violation and calls `sp_hook_dbmlsync_download_log_ri_violation` (if it is implemented). It then calls `sp_hook_dbmlsync_download_ri_conflict` (if it is implemented). If there is still a conflict, dbmlsync deletes the rows. This process is repeated for remaining foreign keys that have RI violations.

This hook is called only when there are RI violations involving tables that are currently being synchronized. If there are RI violations involving tables that are not being synchronized, this hook is not called and the synchronization fails.

This hook is called on the same connection that dbmlsync uses for the download. This hook should not contain any explicit or implicit commits, because they may lead to inconsistent data in the database. The actions of this hook are committed or rolled back when the download stream is committed or rolled back.

Unlike other hook actions, the operations performed during this hook are not

uploaded during the next synchronization.

See also [“sp_hook_dbmlsync_download_log_ri_violation” on page 279](#)

sp_hook_dbmlsync_download_sql_error

Function Handle database errors reading the download stream sent by the MobiLink synchronization server.

Rows in #hook_dict table

Name	Values	Description
table name (in)	table name	The table to which operations were being applied when the error occurred. The value is an empty string if dbmlsync is unable to identify the table.
continue (in out)	True False	Indicates whether the error should be ignored and synchronization should continue. This parameter should be set to true to ignore the error and continue, or false to call the sp_hook_dbmlsync_download_fatal_sql_error hook and stop synchronization. When true is returned in this field the operation that caused the SQL error is lost.
SQL error code (in)	SQL error code	Identifies the SQL error code returned by the database when the operation failed.
publication_n (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one publication_ <i>n</i> entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description If a procedure of this name exists, it is invoked when a database error is detected during the download phase of synchronization. The procedure is

only invoked for errors where it is possible to ignore the error and continue with synchronization. For fatal errors, the `sp_hook_dbmlsync_download_fatal_SQL_error` procedure is called.

Actions of this procedure are committed or rolled back when the download stream is committed or rolled back.

See also

“Synchronization event hook sequence” [*MobiLink Synchronization User’s Guide*, page 194]

[“sp_hook_dbmlsync_download_fatal_sql_error” on page 277](#)

sp_hook_dbmlsync_download_table_begin

Function

Use this stored procedure to add custom actions immediately before each table is downloaded.

Rows in #hook_dict table

Name	Values	Description
table name (in)	table name	The table to which operations are about to be applied.
publication_n (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one publication_ <i>n</i> entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description

If a procedure of this name exists, it is called for each table immediately before downloaded operations are applied to that table. Actions of this procedure are committed or rolled back when the download stream is committed or rolled back.

See also

“Synchronization event hook sequence” [*MobiLink Synchronization User’s Guide*, page 194]

sp_hook_dbmlsync_download_table_end

Function

Use this stored procedure to add custom actions immediately after each table is downloaded.

Rows in #hook_dict table

Name	Values	Description
table name (in)	table name	The table to which operations have just been applied.
delete count (in)	number of rows	The number of rows in this table deleted by the download stream.
upsert count (in)	number of rows	The number of rows in this table updated or inserted by the download stream.
publication_n (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one <code>publication_n</code> entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description If a procedure of this name exists, it is called immediately after all operations in the download stream for a table have been applied.

Actions of this procedure are committed or rolled back when the download stream is committed or rolled back.

See also “Synchronization event hook sequence” [*MobiLink Synchronization User’s Guide*, page 194]

sp_hook_dbmlsync_end

Function Use this stored procedure to add custom actions immediately before synchronization is complete.

Rows in #hook_dict table

Name	Values	Description
restart (in out)	True False	If set to true then, instead of shutting down, dbmlsync begins a new synchronization subject to the same scheduling parameters that applied to the synchronization it just completed. If the field is false (the default) then dbmlsync shuts down or restarts according to its command line arguments.
exit code (in)	number	If set to anything other than zero (the default), this represents a synchronization error.
publication_n (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one publication _n entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
upload status (in)	retry committed failed	<p>Specifies the status returned by the MobiLink synchronization server when dbmlsync attempted to verify receipt of the upload stream.</p> <p>retry The MobiLink synchronization server and dbmlsync had different values for the log offset from which the upload stream should start. The upload stream was not committed by the MobiLink synchronization server. The dbmlsync utility will attempt to send another upload stream starting from a new log offset.</p> <p>committed The upload stream was received by the MobiLink synchronization server, and committed.</p> <p>failed The MobiLink synchronization server did not commit the upload stream.</p>

Name	Values	Description
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description

If a procedure of this name exists, it is called as the last event during synchronization.

Actions of this procedure are committed immediately after execution.

There are cases where dbmlsync never terminates after synchronizing the first publication, so the second will never be synchronized:

If an `sp_hook_dbmlsync_end` hook is defined so that the hook always sets the restart parameter to true, and you specify multiple publications on the dbmlsync command line in the form `-n pub1, -n pub2`, etc., then dbmlsync repeatedly synchronizes the first publication and never synchronizes the second.

See also

“Customizing the client synchronization process” [*MobiLink Synchronization User’s Guide*, page 194]

“Synchronization event hook sequence” [*MobiLink Synchronization User’s Guide*, page 194]

sp_hook_dbmlsync_log_rescan

Function

Use this stored procedure to programmatically decide when a rescan is required.

Rows in #hook_dict table

Name	Values	Description
publication_ <i>n</i> (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one publication_ <i>n</i> entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
discarded storage (in)	number	The number of bytes of discarded memory after the last synchronization.
rescan (in out)	True False	If set to True by the hook, dbmlsync performs a complete rescan before the next synchronization. On entry, this value is set to False.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description

When no other condition has been met that would force a rescan, this hook is called immediately after the `sp_hook_dbmlsync_end` hook. In some cases, this hook may be called even when `dbmlsync` is not hovering, but in these cases, return values from the hook are ignored.

See also [“HoverRescanThreshold \(hrt\) extended option” on page 52](#)

sp_hook_dbmlsync_logscan_begin

Function Use this stored procedure to add custom actions immediately before the transaction log is scanned for upload.

Rows in `#hook_dict` table

Name	Values	Description
starting log offset_ <i>n</i> (in)	number	The log offset value where scanning is to begin. There is one value for each publication being uploaded.
log scan retry (in)	True False	If this is the first time the transaction log has been scanned for this synchronization, the value is false; otherwise it is true. The log is scanned twice when the MobiLink synchronization server and dbmlsync have different information about where the scanning should begin.
publication_ <i>n</i> (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one publication_ <i>n</i> entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description If a procedure of this name exists, it is called immediately before dbmlsync scans the transaction log to assemble the upload stream.

Actions of this procedure are committed immediately after execution.

See also “Synchronization event hook sequence” [*MobiLink Synchronization User’s Guide*, page 194]

sp_hook_dbmlsync_logscan_end

Function Use this stored procedure to add custom actions immediately after the transaction log is scanned for upload.

Rows in #hook_dict table

Name	Values	Description
ending log offset (in)	number	The log offset value where scanning ended.
starting log offset_ <i>n</i> (in)	number	The log offset value where scanning began.
log scan retry (in)	True False	If this is the first time the transaction log has been scanned for this synchronization, the value is false; otherwise it is true. The log is scanned twice when the MobiLink synchronization server and dbmlsync have different information about where the scanning should begin.
publication_ <i>n</i> (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one <i>publication_</i> <i>n</i> entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description If a procedure of this name exists, it is called immediately after dbmlsync has scanned the transaction log to assemble the upload stream.

Actions of this procedure are committed immediately after execution.

See also “Synchronization event hook sequence” [*MobiLink Synchronization User’s Guide*, page 194]

sp_hook_dbmlsync_process_return_code

Function Use this stored procedure to manage return codes.

Rows in #hook_dict table

Name	Values	Description
publication_ <i>n</i> (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one <i>publication_</i> <i>n</i> entry for each publication being uploaded.

Name	Values	Description
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
fatal error (in)	True False	True when the hook is called because of an error that will cause dbmlsync to terminate.
aborted synchronization (in)	True False	True when the hook is called because of an abort request from the sp_hook_dbmlsync_abort hook.
return code (in)	number	The return code from the most recent synchronization attempt. 0 indicates a successful synchronization. Any other value indicates that the synchronization failed. This value can be set by sp_hook_dbmlsync_abort when that hook is used to abort synchronization.
last return code (in)	number	The value stored in the new return code row of the #hook_dict table the last time this hook was called, or 0 if this is the first call to the hook.
new return code (in out)	number	The desired return code for the process. When dbmlsync exits, dbmlsync's return code is the value stored in this row by the last call to the hook. The value must be -32768 to 32767.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description

A dbmlsync session can run multiple synchronizations when you use the -n option, when you use scheduling, or when you use the restart parameter in the end_hook. In these cases, if one or more of the synchronizations fail, the default return code does not indicate which failed. Use this hook to define the return code for the dbmlsync process based on the return codes from the synchronizations. This hook can also be used to log return codes.

Example

Suppose that you run `dbmlsync` to perform five synchronizations and you want the return code to indicate how many of the synchronizations failed, with a return code of 0 indicating that there were no failures, a return code of 1 indicating that one synchronization failed, and so on. You can achieve this by defining the `sp_hook_dbmlsync_process_return_code` hook as follows. In this case, if three synchronizations fail, the new return code is 3.

```
BEGIN
    DECLARE    rc    integer;

    SELECT value INTO rc FROM #hook_dict WHERE name = 'return
        code';
    IF rc <> 0 THEN
        SELECT value INTO rc FROM #hook_dict WHERE name = 'last
            return code';
        UPDATE #hook_dict SET value = rc + 1 WHERE name = 'new
            return code';
        END IF;
    END;
```

See also

“Synchronization event hook sequence” [*MobiLink Synchronization User’s Guide*, page 194]

[“sp_hook_dbmlsync_abort” on page 270](#)

sp_hook_dbmlsync_schema_upgrade

Function

Use this stored procedure to run a SQL script that revises your schema.

Rows in `#hook_dict` table

Name	Values	Description
publication_n (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one <code>publication_n</code> entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
script version	name of script version	The script version that you are synchronizing.

Name	Values	Description
drop hook (out)	never always on success	<p>The values can be:</p> <p>never - (the default) Do not drop the sp_hook_dbmlsync_schema_upgrade hook from the database.</p> <p>always - After attempting to run the hook, drop the sp_hook_dbmlsync_schema_upgrade hook from the database.</p> <p>on success - If the hook runs successfully, drop the sp_hook_dbmlsync_schema_upgrade hook from the database. On success is identical to always if the dbmlsync -eh option is used, or the dbmlsync extended option IgnoreHookErrors is set to true.</p>
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description

This stored procedure is intended for making schema changes to deployed remote databases. Using this hook for schema upgrades ensures that all changes on the remote database are synchronized before the schema is upgraded, which is required to ensure that the database will continue to be able to synchronize. When this hook is being used you should not set the dbmlsync extended option LockTables to off (LockTables is on by default).

During any synchronization where the upload was applied successfully and acknowledged by MobiLink, this hook is called after the sp_hook_dbmlsync_download_end hook and before the sp_hook_dbmlsync_end hook. This hook is not called during download-only synchronization or when a file-based download is being created or applied.

Actions performed in this hook are committed immediately after the hook completes.

See also

“Schema changes in remote databases” [*MobiLink Synchronization User’s Guide*, page 100]

sp_hook_dbmlsync_upload_begin

Function

Use this stored procedure to add custom actions immediately before the transmission of the upload.

Rows in #hook_dict table

Name	Values	Description
publication_n (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one publication_n entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description If a procedure of this name exists, it is called immediately before dbmlsync sends the upload stream.

Actions of this procedure are committed immediately after execution.

See also “Synchronization event hook sequence” [*MobiLink Synchronization User’s Guide*, page 194]

sp_hook_dbmlsync_upload_end

Function Use this stored procedure to add custom actions after dbmlsync has verified receipt of the upload stream by the MobiLink synchronization server.

Rows in #hook_dict table

Name	Values	Description
failure cause (in)	See range of values in Description, below	The cause of failure of an upload. For more information, see Description.

Name	Values	Description
upload status (in)	retry committed failed	<p>Specifies the status returned by the MobiLink synchronization server when dbmlsync attempted to verify receipt of the upload stream.</p> <p>retry The MobiLink synchronization server and dbmlsync had different values for the log offset from which the upload stream should start. The upload stream was not committed by the MobiLink synchronization server. The dbmlsync utility will attempt to send another upload stream starting from a new log offset.</p> <p>committed The upload stream was received by the MobiLink synchronization server, and committed.</p> <p>failed The MobiLink synchronization server did not commit the upload stream.</p>
publication_n (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one publication_ <i>n</i> entry for each publication being uploaded.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.
script version (in)	script version name	The MobiLink script version to be used for the synchronization.

Description

If a procedure of this name exists, it is called immediately after dbmlsync has sent the upload stream and received confirmation of it from the MobiLink synchronization server.

Actions of this procedure are committed immediately after execution.

The range of possible parameter values for the failure cause row in the #hook_dict table includes:

- ◆ **UPLD_ERR_COMMUNICATIONS_FAILURE** A communication error occurred.

- ◆ **UPLD_ERR_LOG_OFFSET_MISMATCH** The upload failed because of conflict between log offset stored on the remote and consolidated databases.
- ◆ **UPLD_ERR_GENERAL_FAILURE** The upload failed for an unknown reason.
- ◆ **UPLD_ERR_INVALID_USERID_OR_PASSWORD** The user ID or password was incorrect.
- ◆ **UPLD_ERR_USERID_OR_PASSWORD_EXPIRED** The user ID or password expired.
- ◆ **UPLD_ERR_USERID_ALREADY_IN_USE** The user ID was already in use.
- ◆ **UPLD_ERR_DOWNLOAD_NOT_AVAILABLE** The upload was committed on the consolidated but an error occurred that prevented MobiLink from generating a download stream.
- ◆ **UPLD_ERR_PROTOCOL_MISMATCH** *Dbmlsync* received unexpected data from the MobiLink synchronization server.
- ◆ **UPLD_ERR_SQLCODE_n** Here, *n* is an integer. A SQL error occurred in the consolidated database. The integer specified is the SQLCODE for the error encountered.

See also

“Synchronization event hook sequence” [*MobiLink Synchronization User’s Guide*, page 194]

sp_hook_dbmlsync_validate_download_file

Function

Use this hook to implement custom logic to decide if a download file can be applied to the remote database.

Rows in #hook_dict table

Name	Values	Description
publication_n (in)	publication name	The publications being synchronized, where <i>n</i> is an integer. There is one publication_n entry for each publication being uploaded. The <i>n</i> in publication_n and generation number_n match.
MobiLink user (in)	MobiLink user name	The MobiLink user for which you are synchronizing.

Name	Values	Description
file last download time (in)		The download file's last download time. (The download file contains all rows that were changed between its last download time and its next last download time.)
file next last download time (in)		The download file's next last download time. (The download file contains all rows that were changed between its last download time and its next last download time.)
file creation time (in)		The time when the download file was created.
file generation number _{<i>n</i>} (in)		There is one file generation number _{<i>n</i>} for each publication _{<i>n</i>} entry. The <i>n</i> in publication _{<i>n</i>} and generation number _{<i>n</i>} match.
user data (in)		The string specified with the dbmlsync -be option when the download file was created.
apply file (in out)	True False	If true (the default), the download file will be applied only if it passes dbmlsync's other validation checks. If false, the download file will not be applied to the remote database.
check generation number (in out)	True False	If true (the default), dbmlsync only applies the download file if the generation numbers stored in the file match those in the remote database. If false, dbmlsync will apply the file even if the generation numbers do not match.

Name	Values	Description
setting generation number (in)	True False	True if the -bg switch was used when the download file was created. If -bg was used, the generation numbers on the remote database are updated from the download file and normal generation number checks are not performed.

Description

Use this stored procedure to implement custom checks to decide if a download file can be applied.

If you want to compare the generation numbers or timestamps contained in the file with those stored in the remote database, they can be queried from the SYSSYNC and SYSPUBLICATION tables.

This hook is called when the -ba option is specified. It is called after the sp_hook_dbmlsync_upload_end hook and before the sp_hook_dbmlsync_download_begin hook.

The actions of this hook are committed immediately after it completes.

See also

[“-be option” on page 41](#)

[“-bg option” on page 41](#)

“File-Based Downloads” [*MobiLink Synchronization User’s Guide*, page 117]

CHAPTER 6

Utilities

About this chapter

This chapter describes the MobiLink utility programs that are required to build and synchronize UltraLite applications.

☞ For information about other Adaptive Server Anywhere utilities, see “Database Administration Utilities” [*ASA Database Administration Guide*, page 455].

Contents

Topic:	page
ActiveSync provider installation utility	300
MobiLink stop utility	303
MobiLink client database extraction utility (deprecated)	304
MobiLink user authentication utility	308
Certificate reader utility	310
Certificate generation utility	311

ActiveSync provider installation utility

Installs a MobiLink provider for ActiveSync, or registers and installs UltraLite applications on Windows CE devices.

Syntax

dbasinst [*options*] [[*src*] *dst name class* [*args*]]

Options	Description
-d	Disable the application on creation.
-k path	Specify the location of the desktop provider <i>dbasdesk.dll</i> .
-n	Register the application but do not copy it to the device.
-u	Uninstall the MobiLink ActiveSync provider.
-v path	Specify the location of the device provider <i>dbasdev.dll</i> .
Args	Description
<i>src</i>	The source filename and path for an application.
<i>dst</i>	The destination filename and path for an application.
<i>name</i>	The name of the application.
<i>class</i>	The registered Windows class name of the application.
<i>args</i>	Command line arguments to use when ActiveSync starts the application.

Description

This utility installs a MobiLink provider for ActiveSync. The provider includes both a component that runs on the desktop (*dbasdesk.dll*) and a component that is deployed to the Windows CE device (*dbasdev.dll*). The *dbasinst* utility makes a registry entry pointing to the current location of the desktop provider; and copies the device provider to the device.

If additional arguments are supplied, the *dbasinst* utility can also be used to register and install UltraLite applications onto a Windows CE device. Alternatively, you can register and install UltraLite applications using the ActiveSync software.

Subject to licensing requirements, you may supply this application, together with the desktop and device components to end users, so that they can prepare their copies of your application for use with ActiveSync.

You must be connected to a remote device to install the ActiveSync provider.

☞ For complete instructions on using the ActiveSync provider installation utility, see “Installing the MobiLink provider for ActiveSync” [*MobiLink*

Synchronization User's Guide, page 223], and “Registering applications for use with ActiveSync” [*MobiLink Synchronization User's Guide*, page 224].

Options

-d By default, an application registered by `dbasinst` is enabled, meaning that it is automatically synchronized when ActiveSync begins a synchronization. With the `-d` option, the application is still registered, but it is unchecked in the ActiveSync MobiLink settings dialog.

-k The path to the desktop provider `dbasdesk.dll`. By default the file is looked for in the `or` or `win64` subdirectory of your SQL Anywhere directory. End users (who generally do not have the full SQL Anywhere install) may need to specify `-k` when installing the MobiLink ActiveSync provider.

-n In addition to installing the MobiLink ActiveSync provider, register an application but do not copy it to the device. This is appropriate if the application includes more than one file (for example, if it is compiled to use the UltraLite runtime library DLL rather than a static library) or if you have an alternative method of copying the application to the device.

-u Unregister all applications that have been registered for use with the MobiLink ActiveSync provider and uninstall the MobiLink ActiveSync provider. No files are deleted from the desktop machine or the device by this operation. If the device is not connected to the desktop, an error is reported.

-v The path to the device provider `dbasdev.dll`. By default the file is looked for in a platform-specific directory under the `CE` subdirectory of your SQL Anywhere directory. End users (who generally do not have the full SQL Anywhere install) may need to specify `-v` when installing the MobiLink ActiveSync provider.

Arguments

src The source filename and path for copying an application to the device. Supply this parameter only if you are registering an application and copying it to the device; do not supply the parameter if you use the `-n` option.

dst The destination filename and path on the device for an application.

name The application name. This is the name by which ActiveSync refers to the application.

class The registered Windows class name for the application.

args Any command line arguments to be used when ActiveSync starts the application.

Examples

The following command installs the MobiLink provider for ActiveSync using default arguments. It does not register an application. The device must be connected to your desktop for the installation to succeed.

```
dbasinst
```

The following command uninstalls the MobiLink provider for ActiveSync. The device must be connected to your desktop for the uninstall to succeed:

```
dbasinst -u
```

The following command installs the MobiLink provider for ActiveSync, if it is not already installed, and registers the application *myapp.exe*. It also copies the *c:\My Files\myapp.exe* file to *\Program Files\myapp.exe* on the device. The *-p -x* arguments are command line options for *myapp.exe* when started by ActiveSync. The command must be entered on a single line:

```
dbasinst "C:\My Files\myapp.exe" "\Program Files\myapp.exe"  
"My Application" MYAPP -p -x
```

See also

“Using ActiveSync synchronization” [*MobiLink Synchronization User’s Guide*, page 189]

“Installing the MobiLink provider for ActiveSync” [*MobiLink Synchronization User’s Guide*, page 223]

“Registering applications for use with ActiveSync” [*MobiLink Synchronization User’s Guide*, page 224]

“Synchronization for UltraLite Applications” [*UltraLite Database User’s Guide*, page 143]

“ActiveSync synchronization stream parameters” [*UltraLite Database User’s Guide*, page 179]

MobiLink stop utility

Stops the MobiLink synchronization server on the local machine.


Syntax

dbmlstop [*options*] [*server-name*]

Option	Description
-f	Forced shutdown. Use if a hard shutdown does not work.
-h	Hard shutdown. MobiLink stops all synchronizations and exits. Some remotes may report an error.
-q	Quiet mode. Suppresses the banner.
-t <i>time</i>	Soft shutdown, with a hard shutdown done after the specified time. <i>time</i> is a number followed by D, H, M, or S (for days, hours, minutes and seconds). For example, -t 10m specifies that the server should be shut down in 10 minutes or when current synchronizations complete, whichever is sooner. D, H, M, and S are not case sensitive.
-w	Waits for the MobiLink synchronization server to shut down before returning from the command.

Parameters

Server-name If the MobiLink synchronization server is started using the **-zs** option, it must be shut down by specifying the same server name.

 For more information, see [“-zs option” on page 30](#).

Description

By default (if none of **-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.

MobiLink client database extraction utility

(deprecated)

Creates an Adaptive Server Anywhere client database using another Adaptive Server Anywhere database as a template. This utility is deprecated. For an alternative way to create client databases, see “Creating a remote database” [*MobiLink Synchronization User’s Guide*, page 168].

Syntax

mlxtract [*additional-options*] *directory site-name*

Option	Description
-ac "keyword=value; ..."	Connect to the database specified in the connect string to do the reload.
-al <i>filename</i>	Log file name for this new database.
-an <i>filename</i>	Creates a database file with the same settings as the database being unloaded and automatically reloads it.
-c "keyword=value; ..."	Supply database connection parameters.
-id	Extract schema definition and data.
-it	Extract triggers.
-j <i>count</i>	Iteration count for view-creation statements.
-l <i>level</i>	Perform all extraction operations at specified isolation level.
-o <i>file</i>	Output messages to file.
-p <i>character</i>	Escape character.
-q	Operate quietly: do not print messages or show windows.
-r <i>file</i>	Specify name of generated reload Interactive SQL command file (default "reload.sql").
-s7	Use Adaptive Server Anywhere version 7 syntax for creating synchronization definitions.
-u	Unordered data.
-v	Verbose messages.
-x	Use external table loads.

Option	Description
-xh	Exclude procedure hooks.
-xf	Exclude foreign keys.
-xp	Exclude stored procedures.
-xv	Exclude views.
-y	Overwrite command file without confirmation.
<i>directory</i>	The directory to which the files are written. This option is not needed if you use -an or -ac .
<i>site-name</i>	Specify which client database to generate.

Description

mlxtract is the MobiLink extraction utility for Adaptive Server Anywhere client databases. It is run against an Adaptive Server Anywhere reference database and creates a new client database or a command file for an Adaptive Server Anywhere client database, depending on the chosen options.

The command line extraction utility creates a command file and a set of associated data files. The command file can be run against a newly initialized Adaptive Server Anywhere database to create the database objects and load the data for the client database.

By default, the command file is named *reload.sql*.

Options

Reload the data to an existing database (-ac) You can combine the operation of extracting a database and reloading the results into an existing database using this option.

For example, the following command (which should be entered all on one line) loads a copy of the data for the *field_user* subscriber into an existing database file named *newdemo.db*:

```
mlxtract -c "uid=DBA;pwd=SQL;dbf=asademo.db" -ac
"uid=DBA;pwd=SQL;dbf=newdemo.db" field_user
```

If you use this option, no copy of the data is created on disk, so you do not specify an unload directory on the command line. This provides greater security for your data, but at some cost for performance.

Reload the data to a new database (-an) You can combine the operations of extracting a database, creating a new database, and loading the data using this option.

For example, the following command (which should be entered all on one line) creates a new database file named *asacopy.db* and copies the schema

and data for the `field_user` subscriber of *asademo.db* into it:

```
mlxtract -c "uid=DBA;pwd=SQL;dbf=asademo.db" -an asacopy.db
        field_user
```

If you use this option, no copy of the data is created on disk, so you do not specify an unload directory on the command line. This provides greater security for your data, but at some cost for performance.

Connection parameters (-c) A set of connection parameters, in a string.

◆ **mlxtract connection parameters** The user ID should have DBA authority to ensure that the user has permissions on all the tables in the database.

For example, the following statement (which should be typed on one line) extracts a database for MobiLink user ID `joe_remote` from the *ASADemo* database running on the `sample_server` server, connecting as user ID `DBA` with password `SQL`. The data is unloaded into the `c:\extract` directory.

```
mlxtract -c "eng=sample_server;dbn=sademo;
uid=DBA;pwd=SQL" c:\extract joe_remote
```

Extract both schema definition and data (-id) By default, only the schema is extracted. Such a database can be initialized with data upon the first connection to a MobiLink synchronization server. This option provides the option of extracting the initial set of data from the reference database.

Extract triggers (-it) By default, triggers are not extracted. This option provides causes triggers present in the reference database to be extracted.

Iteration count for views (-j) If there are nested views in the consolidated database, this option specifies the maximum number of iterations to use when extracting the views.

Perform extraction at a specified isolation level (-l) The default setting is an isolation level of zero. If you are extracting a database from an active server, you should run it at isolation level 3 to ensure that data in the extracted database is consistent with data on the server. Increasing the isolation level may result in large numbers of locks being used by the extraction utility, and may restrict database use by other users.

Output messages to file (-o) Outputs the messages from the extraction process to a file for later review.

Escape character (-p) The default escape character (`\`) can be replaced by another character using this option.

Operate quietly (-q) Display no messages except errors.

Reload filename (-r) The default name for the reload command file is *reload.sql* in the current directory. You can specify a different file name with this option.

Use ASA v7 syntax (-s7) This option is useful when you are using an Adaptive Server Anywhere version 8 or higher consolidated database along with Adaptive Server Anywhere version 7 remote databases. For example, create a version 9 consolidated database, extract the remote databases using the -s7 option, and deploy the reload.sql files to the remote.

Output the data unordered (-u) By default the data in each table is ordered by primary key. Unloads are quicker with the -u option, but loading the data into the client database is slower.

Verbose mode (-v) The name of the table being unloaded and the number of rows unloaded are displayed. The SELECT statement used is also displayed.

Use external loads (-x) In the reload script, the default is to use the LOAD TABLE statement to load the data into the database. If you choose to use external loads, the Interactive SQL INPUT statement is used instead. The LOAD TABLE statement is faster than INPUT.

INPUT takes the path of the data files relative to the client, while LOAD TABLE takes the path relative to the server.

Exclude foreign key definitions (-xf) You can use this if the client database contains a subset of the consolidated database schema, and some foreign key references are not present in the client database.

Exclude stored procedure (-xp) Do not extract stored procedures from the database.

Exclude views (-xv) Do not extract views from the database.

Operate without confirming actions (-y) Without this option, you are prompted to confirm the replacement of an existing reloadcommand file.

MobiLink user authentication utility

Registers MobiLink users at the consolidated database. For Adaptive Server Anywhere remotes, the users must have previously been created at the remote databases with the CREATE SYNCHRONIZATION USER statement.

Syntax

dbmluser [*options*] **-c** "*connection-string*"
{ **-f** *file* | **-u** *user* [**-p** *password*] }

Option	Description
-c " <i>keyword=value;...</i> "	Supply database connection parameters. The connection string must give the utility permission to connect to the consolidated database using an ODBC data source. This parameter is required.
-d	Deletes the user name(s) specified by -f or -u .
-dl	Display messages in the window or on the command line and also in the log file, if specified.
-f <i>filename</i>	Read the user names and passwords from the specified file. The file should be a text file containing one user name and password pair on each line, separated by white space. You must specify either -f or -u .
-o <i>filename</i>	Log output messages to the specified file.
-os <i>size</i>	Limit the size of the output file. The <i>size</i> is the maximum file size for logging output messages, specified in bytes. Use the suffix k or m to specify units of kilobytes or megabytes, respectively. By default, there is no size limit. The minimum size limit is 10 kb.
-ot <i>filename</i>	Truncate the log file and then append output messages to it. The default is to send output to the screen.
-p <i>password</i>	Password to associate with the user. This option can only be used with -u .

Option	Description
-pc <i>collation-id</i>	Supply a database collation ID for character set translation of the user name and password. This should be one of the Adaptive Server Anywhere collation labels such as those listed in “Initialization utility options” [ASA Database Administration Guide, page 487]. For machines using single-byte character sets the default is 1252LATIN1 . For machines using multi-byte character sets, the default is 932JPN .
-u <i>ml_username</i>	Specify the user name to add (or delete, if used with -d). Only one user can be specified on a single command line. This option is used with -p if passwords are being used. You must specify either -f or -u.

Description

Given a user/password pair, the dbmluser utility first attempts to add the user. If the user has already been added to the consolidated database, it attempts to update the password for that user.

There are alternative ways to register user names in the consolidated database:

- ◆ Use Sybase Central.
- ◆ Specify the -zu+ command line option with dbmlsrv9. In this case, any existing MobiLink users that have not been added to the consolidated database are added when they first synchronize.

The MobiLink user must already exist in a remote database. To add users at the remote, you have the following options:

- ◆ For Adaptive Server Anywhere remotes, set the name with CREATE SYNCHRONIZATION USER and synchronize with that user name.
- ◆ For UltraLite remotes, you can either use the user_name field of the ul_synch_info structure; or in Java, use the SetUserName() method of the ULSynchInfo class before synchronizing.

See also

“Authenticating MobiLink Users” [MobiLink Synchronization User’s Guide, page 103]

Certificate reader utility

Use the *readcert* utility to display values within a certificate and validate the chain of certificates.

Syntax

readcert *certificate-name*

Description

The certificate you specify can be elliptic-curve or RSA.

When synchronization occurs through an ECC_TLS or RSA_TLS synchronization stream, the MobiLink synchronization server sends its certificate to the client, as well as the certificate of the entity that signed it, and so on up to a self-signed root. The client checks that the chain is valid and that it trusts the root certificate in the chain.

This utility scans an X509 authentication certificate and displays the field values. It then checks that the chain of certificates is valid. A validation error is reported if any of the certificates in the chain have expired, are in the wrong order, or are missing.

See also

“Transport-Layer Security” [*MobiLink Synchronization User’s Guide*, page 337]

Certificate generation utility

Use the gencert utility to create a new elliptic-curve or RSA certificate, or to sign a pre-generated certificate request.

☞ For more information about security of MobiLink synchronization, see “Transport-Layer Security” [*MobiLink Synchronization User’s Guide*, page 337].

Syntax

gencert [**-c** | **-s**] [**-r**] [**-q**]

Option	Description
-c	Generate a certificate authority certificate.
-q <i>request-file</i>	Sign a pre-generated certificate request.
-r	Generate a self-signed root certificate.
-s	Generate a server identity certificate.

Description

This utility creates a new X509 certificate. When first started, it prompts whether you want to generate an elliptic-curve or RSA certificate.

If you are generating an elliptic-curve certificate, gencert generates an elliptic-curve key pair. If you are generating an RSA certificate, it prompts for a key size between 512 and 2028, and then creates a certificate using RSA.

The gencert utility then requests values for the distinguished fields. These fields include the country, state or province, locality, organization, organizational unit, and common name, the serial number, and an expiry date. It then requests the file name of a certificate that is to sign the new certificate.

If no certificate name is supplied, the new certificate becomes a root certificate. If a certificate file name is supplied, gencert reads and validates the certificate chain and requests the name of the file that contains the signer’s private key. It then requests the password for that private key.

Then the utility requests the password that is to protect the new private key.

This utility writes three different types of files. One file contains only the new certificate. Another contains only the encrypted private key, and a third file contains both the certificate and the encrypted private key.

Often, not all three files are needed. For example, if the certificate is to be a certificate authority, used to sign other certificates, the file that contains only the certificates is distributed as a trusted root certificate to clients. The file containing the encrypted private key is stored securely. In this case, security

is improved by storing the private key and the certificate separately, so the third file is not generated.

If, instead, the certificate is to identify a server, the encrypted private key should be stored with the certificate, so the utility writes only the file that contains both pieces of information.

If the signing certificate is not a root certificate, but is instead part of a chain, *gencert* reads and validates the entire chain before issuing the new certificate.

When generating a server identity certificate, the entire chain is always saved. Otherwise, saving the entire chain is optional.

When signing a pre-generated certificate request, *gencert* only prompts for a serial number, expiry date, the certificate and private key of the signer, and an output file for the signed certificate.

Gencert can sign any request that is generated by the Certicom *reqtool* utility or any other third party application that generates certificate requests in the appropriate format, such as the Microsoft IIS Web server or the Netscape iPlanet Web server. Following is an example of a certificate that is in the appropriate format:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBqjCCARMCAQAwjELMAkGA1UEBhMCVDALBgNVBAgTBHRlc3QxDTALBgNV
BAcTBHRlc3QxDTALBgNVBAoTBHRlc3QxDBgNVBAcTBHRlc3QxH2AdBgNVBAMT
Fm12YW5kZkZwLXBjLnN5YmFzZS5jb20wgDQYJKoZIhvcNAQEBBQADgY0AMIGJ
AoGBAKad6a15MDIGYNGO1ctjAeFl6VSVglg1z1OEMILjyAW51zDMJo1KFazxc
ptGs0AlKbJH/1EHUeJ4kp7zGuyV4OipEw9NSxzza6mSKewsulR735CY8X07Z/
agfa3NGRiYEC39/SD3+bcN7NkDn250xJ6Yxbfcf/1EUTNagMBAAGgADANBgkq
hkiG9w0BAQQFAAOBgQAvgnKRtSVLEUFIQ/abo959UBf+ZDoZzUCx1vnkUjBrA
G/zVDu2A3rqazsr17ihP0nRWnr+iFj+vK2Lg6jiFAzBxC/3w3fWYYJ6ImvodX
coYD3EuoXxWcKfIRq6AAB8SlJcdjntz8HXmWm2tNXVUIcXuEz00ErANOPXQ==
-----END NEW CERTIFICATE REQUEST-----
```

Options

-c Generate a certificate authority certificate. A certificate authority can be used to sign other certificates. By default, generated certificates cannot be used as certificate authorities.

-q Sign a pre-generated certificate request. You can specify either an elliptic-curve or an RSA certificate to be signed.

-r Generate a root certificate. A root certificate is signed only by itself. The default is to prompt for the name of a file that contains the certificate that is to sign the new, generated certificate.

-s Generate a server identity certificate, used to identify a MobiLink synchronization server, rather than a client. A server identity certificate cannot be a certificate authority, so this option is incompatible with the **-c**

option.

See also

“Transport-Layer Security” [*MobiLink Synchronization User’s Guide*, page 337]

Example

The following example signs a certificate request called *certreq.txt*.

```
c:\>gencert -s -q certreq.txt
Certificate Generation Tool
Serial Number: 01
Certificate valid for how many years: 10
Enter file path of signer’s certificate: rsaroot.crt
Enter file path of signer’s private key: rsaroot.key
Enter password for signer’s private key: test
Enter file path to save certificate: testcert.crt
Save entire chain (y/n): y
```


CHAPTER 7

MobiLink System Tables

About this chapter

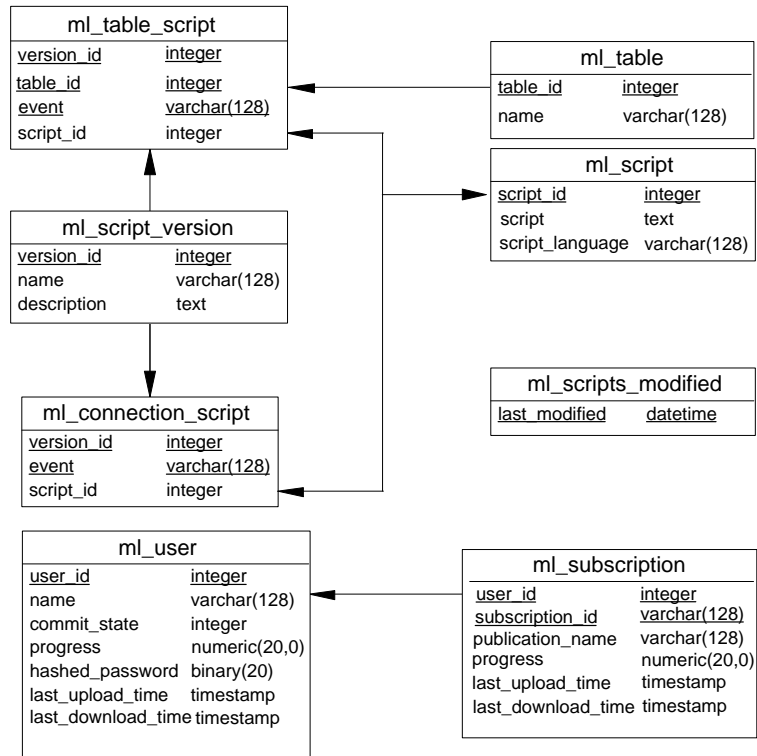
This chapter describes the MobiLink administration tables.

Contents

Topic:	page
Introduction	316

Introduction

The MobiLink system tables store information about MobiLink users, subscriptions, tables, scripts, and script versions. They are created when you run the MobiLink setup scripts for your consolidated database, and are updated as you use MobiLink. They are not like RDBMS system tables in that you can alter them directly.



☞ For more information about how to create these tables, see “Setting up a consolidated database” [*MobiLink Synchronization User’s Guide*, page 11].

The MobiLink system tables are stored in the consolidated database. Unlike most system tables, you can modify them. However, except in rare circumstances you should not alter them directly.

Adaptive Server Anywhere remote databases also have system tables. For more information, see “System Tables” [*ASA SQL Reference*, page 611]. UltraLite databases do not have system tables.

Notes

- ◆ The following section shows details of the MobiLink system tables. In some DBMSs, the data types are slightly different.

- ◆ IBM DB2 UDB version 5.2 only supports column names and other identifiers of 18 characters or less. In a DB2 5.2 consolidated database, MobiLink system tables are truncated where necessary.

ml_connection_script

For a given script version, associates a script with a given event.

Row	Description
version_id	INTEGER. Primary key. The version of the connection script. The version_id column references the version_id column of the ml_script_version table.
event	VARCHAR(128). Primary key. The event column stores the name of the event that triggers the connection script.
script_id	INTEGER. Foreign key. The script_id column references the script_id column of the ml_script system table. The text of the connection script is stored in the ml_script system table.

ml_script

Stores the text of all scripts.

Row	Description
script_id	INTEGER. Primary key. The script_id column stores a unique integer that identifies the script.
script	TEXT. The script column stores the text of the script.
script_language	VARCHAR(128). The script_language column stores the scripting language used for the script. The scripting language can be sql , java , or dnet .

ml_script_version

Stores the name, ID and comment of script associated with each script version.

Row	Description
version_id	INTEGER. Primary key. The version_id column stores a unique integer that identifies the version.
name	VARCHAR(128). The name column stores the arbitrary name given to the version.
description	TEXT. The description column stores the arbitrary description given to the version. The description is not used by MobiLink, but is useful for application-specific comments. For example, you could describe the purpose of a given script version.

ml_scripts_modified

Stores the last time script tables were changed. MobiLink server checks this table to determine if it must load new scripts.

Row	Description
last_modified	DATETIME. Primary key. The last_modified column stores the last time when the ml_script_version, ml_script or ml_connection_script system table was altered.

ml_subscription

Keeps track of the log offsets per subscription for Adaptive Server Anywhere remote databases.

Row	Description
user_id	INTEGER. Primary key. The user_id column references the user_id column of the ml_user table. This is the ID of a user who has subscribed to the publication.
subscription_id	VARCHAR(128). Primary key. The subscription_id is a number that is generated by Adaptive Server Anywhere on the remote.
publication_name	VARCHAR(128). Primary key. The publication_name stores the user-defined name for the publication.
progress	NUMERIC(20,0). The progress, also called the offset, refers to the point in the remote transaction log up to which all operations for the subscription have been uploaded. This column is used for version 8.0 and up databases. For version 7 databases, the progress is stored in the ml_user table.
last_download_time	TIMESTAMP. Indicates the last time a download was applied to the consolidated for a given user or subscription. The default is January 1, 1900, 00:00:00.
last_upload_time	TIMESTAMP. Indicates the last time an upload was applied to the consolidated for a given user or subscription. The default is January 1, 1900, 00:00:00.

ml_table

Stores names of remote tables. This list includes any table marked as a synchronized table in Sybase Central.

Row	Description
table_id	INTEGER. Primary key. The table_id column stores a unique integer identifying the table.
name	VARCHAR(128). The name column stores the arbitrary name given to the table.

ml_table_script

For a given script version, associates a table script with a given table and event.

Row	Description
version_id	INTEGER. Primary key. The version_id column references the version_id column of the ml_script_version table.
table_id	INTEGER. Primary key. The table_id column references the table_id column of the ml_table system table.
event	VARCHAR(128). Primary key. The event column stores the name of the event.
script_id	INTEGER. The script_id column references the script_id column of the ml_script table. The script is stored in the ml_script table.

ml_user

Stores all registered MobiLink users, including their password and their synchronization state. The state is used only for UltraLite remotes or version 7 Adaptive Server Anywhere remotes.

Row	Description
user_id	INTEGER. Primary key. The user_id column stores a unique integer identifying the user. This value is only used internally by MobiLink.
name	VARCHAR(128). The name column stores the arbitrary name given to the user.
commit_state	INTEGER. The commit_state column stores the state.
progress	NUMERIC(20,0). The progress, also called the offset, refers to the point in the transaction log up to which all operations for subscriptions have been uploaded and acknowledged. This column is used for version 7 databases. For version 8.0 and up databases, the progress is stored in the ml_subscription table.
hashed_password	BINARY(20). The hashed_password column stores the MobiLink user's password in obfuscated form. If there is no password, this value is NULL. (This is not recommended.)
last_download_time	TIMESTAMP. Indicates the last time a download was applied to the consolidated for a given user or subscription. The default is January 1, 1900, 00:00:00.
last_upload_time	TIMESTAMP. Indicates the last time an upload was applied to the consolidated for a given user or subscription. The default is January 1, 1900, 00:00:00.

CHAPTER 8

Data Type Conversions

About this chapter

This chapter provides information about the conversion of data types that must take place when a MobiLink synchronization server communicates with a consolidated database that is not Adaptive Server Anywhere. The following tables identify these conversions.

If you are writing synchronization scripts in .NET languages or in Java, you may need to know how to map SQL data types to Java and .NET data types. For more information, see “SQL-.NET data types” [*MobiLink Synchronization User's Guide*, page 261] and “SQL-Java data types” [*MobiLink Synchronization User's Guide*, page 233].

Note

Only supported data types are presented in this chapter.

Contents

Topic:	page
Sybase Adaptive Server Enterprise	324
IBM DB2	326
Oracle	328
Microsoft SQL Server	330

Sybase Adaptive Server Enterprise

The following table identifies how Adaptive Server Anywhere data types are mapped to Adaptive Server Enterprise data types.

Adaptive Server Anywhere data type	Adaptive Server Enterprise data type
bit	bit
tinyint	tinyint
smallint	smallint
int	int
integer	integer
decimal [defaults p=30, s=6]	numeric(30,6)
numeric [defaults p=30 s=6]	numeric(30,6)
float	real
real	real
double	float
smallmoney	numeric(10,4)
money	numeric(19,4)
date	datetime
time	datetime
timestamp	datetime
smalldatetime	datetime
datetime	datetime
char(n)	varchar(n)
character(n)	varchar(n)
varchar(n)	varchar(n)
character varying(n)	varchar(n)
long varchar	text
text	text
binary(n)	binary(n)

Adaptive Server Anywhere data type	Adaptive Server Enterprise data type
long binary	image
image	image
bigint	numeric(20,0)

IBM DB2

The following table identifies how Adaptive Server Anywhere data types are mapped to IBM DB2 data types.

Adaptive Server Anywhere data type	IBM DB2 data type
bit	smallint
tinyint	smallint
smallint	smallint
int	int
integer	int
bigint	decimal(20,0)
char(1–4000)	varchar(n)
char(4001–32767)	long varchar
character(1–4000)	varchar(n)
character(4001–32767)	long varchar
varchar(1–4000)	varchar(n)
varchar(4001–32767)	long varchar
character varying(1–4000)	varchar(n)
character varying(4001–32767)	long varchar or CLOB(n)
long varchar	long varchar or CLOB(n)
text	long varchar
binary(1–4000)	varchar for bit data or BLOB(n)
binary(4001–32767)	long varchar for bit data or BLOB(n)
long binary	long varchar for bit data or BLOB(n)
image	long varchar for bit data or BLOB(n)
decimal [defaults p=30, s=6]	decimal(30,6)
numeric [defaults p=30 s=6]	decimal(30,6)
real	real
float	float

Adaptive Server Anywhere data type	IBM DB2 data type
double	float
smallmoney	decimal(10,4)
money	decimal(19,4)
date	date
time	time
smalldatetime	timestamp
datetime	timestamp
timestamp	timestamp

Oracle

The following table identifies how Adaptive Server Anywhere data types are mapped to Oracle data types.

Adaptive Server Anywhere data type	Oracle data type
bit	number(1,0)
tinyint	number(3,0)
smallint	number(5,0)
int	number(11,0)
bigint	number(20,0)
decimal(prec, scale)	number(prec, scale)
numeric(prec, scale)	number(prec, scale)
float	float
real	real
smallmoney	numeric(10,4)
money	number(19,4)
date	date
time	date
timestamp	date
smalldatetime	date
datetime	date
char(n)	varchar(n) or CLOB(n)
varchar(n)	varchar(n) or CLOB(n)
long varchar	CLOB
binary(n)	raw(n) or BLOB(n)
varbinary(n)	raw(n) or BLOB(n)
long binary	BLOB

The LONG data types are deprecated in Oracle 8, 8i and 9i.

For Oracle LONG data types to synchronize properly, you must check the Oracle **Force Retrieval of Long Columns** ODBC option in the ODBC data source configuration dialog.

Microsoft SQL Server

The following table identifies how Adaptive Server Anywhere data types are mapped to Microsoft SQL Server data types.

Adaptive Server Anywhere data type	Microsoft SQL Server data type
bit	bit
tinyint	tinyint
smallint	smallint
int	int
bigint	numeric(20,0) or bigint (SQL Server 2000 only)
decimal [defaults p=30, s=6]	decimal(30, 6)
numeric [defaults p=30 s=6]	numeric(30, 6)
float	float
real	real
smallmoney	smallmoney
money	money
date	datetime
time	datetime
timestamp	datetime
smalldatetime	datetime
datetime	datetime
char(n)	varchar(n) or text
character(n)	varchar(n)
varchar(n)	varchar(n) or text
long varchar	text
binary(n)	binary(n) or image
long binary	image
double	float

CHAPTER 9

Character Set Considerations

About this chapter

This chapter describes how to handle international language issues in MobiLink applications.

Contents

Topic:	page
Character set considerations	332

Character set considerations

Each character of text is represented in one or more bytes. The mapping from characters to binary codes is called the **character set encoding**. Some character sets used for languages with small alphabets, such as European languages, use a single-byte representation. Others, such as Unicode, use a double-byte representation. Because they use twice the storage space for each character, double-byte character sets can represent a much larger number of characters.

Conversion errors can occur or data can be lost when text using one character set must be translated to another character set. Not all characters can be represented in all character sets. In particular, single-byte character sets can represent a much smaller number of characters than multi-byte systems because of the limited number of codes available.

When the character set of your MobiLink remote database is the same as your consolidated database, character translation issues are avoided.

Text often needs to be sorted to build indexes and to prepare ordered result sets, such as directory listings. The **sort order** identifies the order of the characters. For example, a sort order typically states that the letter “a” comes before the letter “b”, which comes before the letter “c”.

Each database has a **collation sequence**. You set the collation sequence when you create the database, although how you do so can differ between database systems. The collation sequence defines both the character set and the sort order for that database.

Tip

Whenever possible, define the collation sequence of your remote database to be the same as that of your consolidated database. This arrangement reduces the chance of erroneous translations.

☞ For more information, see “Character sets in UltraLite” [*UltraLite Database User’s Guide*, page 40] and “International Languages and Character Sets” [*ASA Database Administration Guide*, page 285].

Character-set translation during synchronization: Windows

During synchronization, characters may need to be translated from one character set to another. The following translations occur as characters are passed between the remote application and the consolidated database.

Character-set translation during upload

The MobiLink client sends data to the MobiLink synchronization server using the character set of the remote database.

	<ol style="list-style-type: none"> 1. The MobiLink synchronization server communicates with the consolidated database using the Unicode ODBC API. To do so, the MobiLink synchronization server translates all characters received from the remote database into Unicode. 2. If necessary, the ODBC driver for the consolidated database server translates the characters from Unicode into the character set of your consolidated database. This translation is controlled solely by the ODBC driver for your consolidated database system. Hence, behavior can differ between two different database systems, particularly systems made by different manufacturers. MobiLink synchronization works with a number of database systems. Check the documentation of your particular consolidated server and ODBC driver for details.
Character-set translation during download	<ol style="list-style-type: none"> 1. The ODBC driver for the consolidated database system receives characters in the coding of the consolidated database. It translates these characters into Unicode to pass them through the Unicode API to the MobiLink synchronization server. This translation is controlled solely by the ODBC driver for your consolidated database system. Check the documentation of your particular consolidated server and ODBC driver for details. 2. The MobiLink synchronization server receives characters through the Unicode ODBC API. If the remote database uses a different character set, the MobiLink synchronization server translates the characters before downloading them.
Examples	<ul style="list-style-type: none"> ◆ UltraLite applications on Windows CE devices use the Unicode character set. When you synchronize a Windows CE application, no character translation occurs within the MobiLink synchronization server. The server finds that data arriving from the application is already in Unicode and passes it directly to the ODBC driver. Similarly, no character-set translation is necessary when downloading data. ◆ All Adaptive Server Anywhere databases and all UltraLite applications on platforms other than Windows CE use the character set determined by the collating sequence of the remote database. When you synchronize a remote database, the MobiLink synchronization server performs character set translations between the character set of the remote database and Unicode.

Controlling ODBC driver character-set translation

Because most consolidated databases are unlikely to use Unicode, it is important to understand how the ODBC driver for your consolidated

database system converts data to and from Unicode. Some ODBC drivers use the language settings of the machine running MobiLink to determine what character set to use. In these cases, it is best if the language and code-page settings of the machine running the MobiLink synchronization server match those of the consolidated database.

Other ODBC drivers, such as the driver for Sybase Adaptive Server Enterprise, allow each connection to use a specific character set. To avoid translation errors, the character set used by MobiLink should be set to match that of the consolidated database.

☞ For a detailed description of how character-set translations take place in your consolidated database server's ODBC driver, consult that product's ODBC driver documentation.

Character set translation during synchronization: non-Windows

The ODBC drivers that iAnywhere Solutions provides on non-Windows platforms do not have a Unicode ODBC API. The MobiLink synchronization server exchanges data with the ODBC driver using the character set determined by the collating sequence of the remote database.

When the remote database is an UltraLite application running under Windows CE, the MobiLink synchronization server performs character-set translation between Unicode and the character set being used with ODBC.

CHAPTER 10

ODBC Drivers

About this appendix

This appendix describes the ODBC drivers available for use with MobiLink.

Contents

Topic:

page

[ODBC drivers supported by MobiLink](#)

[336](#)

ODBC drivers supported by MobiLink

The MobiLink synchronization server can work with a variety of consolidated databases and ODBC drivers, as shown in the table below. Some drivers, though compatible for use with MobiLink, may have functional restrictions associated with their use.

For updated information and complete functional specifications, see http://www.ianywhere.com/developer/technotes/odbc_mobilink.html.

☞ For information about configuring ODBC drivers, see “Introduction to ODBC Drivers” [*iAnywhere Solutions ODBC Drivers*, page 1]. On UNIX, a standalone version of this book is installed in the *drivers* subdirectory of your SQL Anywhere Studio install directory.

Database	ODBC Drivers
Oracle 8i	iAnywhere Solutions 8 - Oracle 8, 8i & 9i ODBC Driver Merant DataDirect Connect ODBC Driver for Oracle
Oracle 9i	iAnywhere Solutions 8 - Oracle 8, 8i & 9i ODBC Driver Oracle 9i ODBC Driver
Microsoft SQL Server 7, Microsoft SQL Server 2000	Microsoft SQL Server ODBC Driver Merant DataDirect SQL Server Wire Protocol ODBC Driver
Sybase Adaptive Server Enterprise 11.5 or later	iAnywhere Solutions 8 - Sybase ASE ODBC Driver Sybase ASE ODBC Driver Merant DataDirect Sybase Wire Protocol ODBC Driver Merant DataDirect Connect ODBC Driver for Sybase ASE Merant Connect ODBC Driver for Sybase ASE
IBM DB2 UDB 7.1, 7.2	IBM DB2 UDB 7.1 ODBC driver IBM DB2 UDB 7.2 ODBC driver Merant DataDirect DB2 Wire Protocol ODBC Driver
Sybase Adaptive Server Anywhere 9	Adaptive Server Anywhere 9.0 ODBC Driver

CHAPTER 11

Deploying MobiLink Applications

About this appendix

This appendix describes how to deploy the MobiLink server and MobiLink clients in a production environment. It identifies the files required for deployment.

Check your license agreement

Redistribution of files is subject to your license agreement. No statements in this document override anything in your license agreement. Please check your license agreement before considering deployment.

Contents

Topic:	page
Deployment overview	338
Deploying the MobiLink server	339
Deploying Adaptive Server Anywhere MobiLink clients	342
Deploying UltraLite MobiLink clients	344

Deployment overview

Deploying MobiLink applications involves the following activities:

- ◆ Deploy the MobiLink server into a production setting.
- ◆ Deploy any Adaptive Server Anywhere MobiLink clients.
- ◆ Deploy any UltraLite MobiLink clients.

This chapter describes the files you need to include in your application's install program for each of these items.

Deploying the MobiLink server

The simplest way to deploy a MobiLink synchronization server into a production environment is to install a licensed copy of SQL Anywhere Studio onto the production machine.

However, if you are redistributing a MobiLink synchronization server in a separate install program (subject to your license agreement), you may include only a subset of the files. In this case, you need to include the following files in your installation.

Notes

- ◆ Test on a clean machine before redistributing.
- ◆ Files must be installed within the Adaptive Server Anywhere installation under your deployment location.
- ◆ The files should be in the same directory, unless otherwise mentioned.
- ◆ When a location is given, the files must be copied into a directory of the same name.

Windows applications

Description	Windows files
MobiLink synchronization server	<i>dbmlsrv9.exe, dbmlsv9.dll, dbunic9.dll, dbmsql9.dll, charsets\unicode</i>
Language library	<i>dblgcn9.dll¹</i>
Windows Performance Monitor support	<i>dbmlctr9.dll², dbmlctr9.ini, dbmlctr9.h</i>
Synchronization stream libraries (deploy the ones you use)	<i>dbmlsock9.dll, dbmlhttp9.dll, dbmlhttps9.-dll, dbmlrsa9.dll</i>
Java synchronization logic	<i>win32\dbmjava9.dll, win32\mljodbc9.-dll, java\mlscript.jar, java\jodbc.-jar, java\mlnotif.jar, java\mailapi.jar, java\smtp.jar, java\activation.jar</i>
.NET synchronization logic	<i>win32\dnetodbc9.dll, win32\iAnywhere.-MobiLink.dll, win32\iAnywhere.-MobiLink.Script.dll, win32\dbmdnet9.-dll, win32\mlDomConfig.xsd, win32\iAnywhere.MobiLink.Script.xml, MobiLink\setup\dnet\mlDomConfig.xml</i>

Description	Windows files
Security option (separately licensable)	<i>win32\dbmltls9.dll, win32\dbmljtls9.dll</i>
Script files (deploy the ones for your consolidated database)	<i>MobiLink\setup, MobiLink\upgrade</i>
iAnywhere Solutions ODBC drivers	<i>\drivers</i>
dbmluser utility	<i>dbmluser.exe</i>
dbmlstop utility	<i>dbmlstop.exe</i>
error names	<i>h\sserror.h</i>
MobiLink Monitor	<i>java\mlmon.jar, dbmlmon.exe, shared\java\jsyblib14.jar, shared\java\HelpManager11.jar, ultralite\java\lib\ulrt.jar</i>
language jar	<i>java\dbmaen9.jar</i>
MobiLink Redirector	<i>MobiLink\redirector</i>
Sybase Central	<i>shared\Sybase Central 4.2</i>
Sybase Central plug-in	<i>win32\dbmlput9.dll</i>

Note: Files may be located in the *win32* or *win64* directory, depending on your version of the software.

¹ For French, German, Japanese, and Chinese editions, substitute **en** with **fr**, **de**, **ja**, and **zh**, respectively.

² Your setup program must self-register this file.

UNIX applications

Description	UNIX files
MobiLink synchronization server	<i>dbmlsrv9, lib/libdbunic9_r.so, lib/libdbmsql9_r.so, lib/libdbtasks9_r.so, charsets/unicode</i>
Language library	<i>dblg9.res¹</i>
Windows Performance Monitor support	N/A

Description	UNIX files
Synchronization stream libraries (deploy the ones you use)	<i>lib/libdbmlsock9_r.so, lib/libdbmlhttp9_r.so, lib/libdbmlhttps9_r.so, lib/libdbmlrsa9_r.so</i>
Java synchronization logic	<i>lib/libdbmjava9_r.so, lib/libmljodbc9.so, java/mlscript.jar, java/jodbc.jar, java/mlnotif.jar, java/mailapi.jar, java/smtp.jar, java/activation.jar</i>
.NET synchronization logic	N/A
Security option (separately licensable)	<i>lib/libdbmltls9_r.so, lib/dbmljtls9_r.so</i>
Script files (deploy the ones for your consolidated database)	<i>MobiLink/setup, MobiLink/upgrade</i>
iAnywhere Solutions ODBC drivers	<i>drivers</i>
dbmluser utility	<i>dbmluser</i>
dbmlstop utility	<i>dbmlstop</i>
error names	<i>h/sserror.h</i>
MobiLink Monitor	<i>java/mlmon.jar, bin/dbmlmon, shared/java/jsyblib14.jar, shared/java/HelpManager11.jar, ultralite/java/lib/ulrt.jar</i>
language jar	<i>java/dbmaen9.jar</i>
MobiLink Redirector	<i>redirector/redirector.config, redirector/java</i>
Sybase Central	<i>shared/sybcentral42</i>
Sybase Central plug-in	<i>dbmlput9.so</i>

¹ For French, German, Japanese, and Chinese editions, substitute **en** with **fr**, **de**, **ja**, and **zh**, respectively.

Deploying Adaptive Server Anywhere MobiLink clients

For Adaptive Server Anywhere clients, you need to deploy an Adaptive Server Anywhere database server and the MobiLink client.

☞ For information about deploying Adaptive Server Anywhere databases, see “Deploying Databases and Applications” [*ASA Programming Guide*, page 467].

If you are redistributing MobiLink synchronization clients (subject to your license agreement) you need to include the following files in your installation in addition to those required for the Adaptive Server Anywhere database:

The files should be in the same directory, unless otherwise mentioned.

Windows applications

Description	Windows files
MobiLink synchronization client	<i>dbmlsync.exe</i> , <i>dbtool9.dll</i> , <i>lib/libdbtasks9.so</i> , <i>lib/libdbtasks9_-</i> <i>r.so</i> , <i>dblggen9.dll</i>
Synchronization stream libraries (deploy the ones you use)	<i>lib/libdbmlsock9.so</i> , <i>lib/libdbmlsock9_r-</i> <i>so</i> , <i>lib/libdbmlhttp9-</i> <i>so</i> , <i>lib/libdbmlhttp9_r-</i> <i>so</i> , <i>lib/libdbmlhttps9-</i> <i>so</i> , <i>lib/libdbmlhttps9_r.so</i> , <i>lib/libdbmlrsa9.so</i> , <i>lib/libdbmlrsa9_r-</i> <i>so</i>
Security option (separately licens- able)	<i>dbmltls9.dll</i>

UNIX applications

Description	UNIX files
MobiLink synchronization client	<i>dbmlsync</i> , <i>lib/libdbtool9.so</i> , <i>lib/libdbtool9_r.so</i> , <i>dbngen9.res</i>
Synchronization stream libraries (deploy the ones you use)	<i>lib/libdbmlsock9.so</i> , <i>lib/libdbmlsock9_r-</i> <i>so</i> , <i>lib/libdbmlhttp9-</i> <i>so</i> , <i>lib/libdbmlhttps9_r.so</i> , <i>lib/libdbmlrsa9.so</i> , <i>lib/libdbmlrsa9_r-</i> <i>so</i>
Security option (separately licens- able)	<i>lib/libdbmltls9.so</i> , <i>lib/libdbmltls9_r.so</i>

Deploying UltraLite MobiLink clients

For UltraLite clients, the UltraLite runtime library or the UltraLite component includes the required synchronization stream functions. The UltraLite runtime library is compiled into your application. Deployment is subject to your license agreement.

☞ For more information, see the documentation for the UltraLite component or development model you are using.

PART II

ERROR AND WARNING MESSAGES

This part describes MobiLink error and warning messages.

CHAPTER 12

MobiLink Communication Error Messages

About this chapter

This chapter lists MobiLink client/server communication errors, as well as their probable causes.

The error messages are written to the MobiLink synchronization server message log and the MobiLink Adaptive Server Anywhere client message log. The error codes are returned to UltraLite clients in the **ss_error_code** member of the **stream_error** parameter.

Contents

Topic:	page
Communication error messages sorted by code	348
Communication error messages sorted by message	352
Communication error messages sorted by constant	356
Communication error descriptions	362

Communication error messages sorted by code

Error code	Error message
0	"No error or unknown error." on page 370
1	"Invalid parameter '%1!s!.'" on page 370
2	"Parameter value '%1!s!' is not an unsigned integer." on page 371
3	"Parameter value '%1!s!' is not an unsigned integer value or range. A range has the form NNN-NNN." on page 372
4	"Parameter value '%1!s!' is not a valid boolean value. The value must be 0 or 1." on page 371
5	"Parameter value '%1!s!' is not a valid hexadecimal value." on page 371
6	"Unable to allocate %1!s! bytes." on page 369
7	"Unable to parse the parameter string '%1!s!.'" on page 372
8	"Unable to read %1!s! bytes." on page 373
9	"Unable to write %1!s! bytes." on page 396
10	"An end write failed." on page 363
11	"An end read failed." on page 363
12	"Feature not implemented." on page 370
13	"The operation would cause blocking." on page 396
14	"Unable to generate a random number." on page 364
15	"Unable to initialize the random number generator." on page 369
16	"Unable to seed the random number generator." on page 387
17	"Unable to create a random number object." on page 362
18	"An error occurred during shutdown." on page 388
19	"Unable to dequeue from the connection queue." on page 363
20	"Invalid root certificate." on page 380

Error code	Error message
21	"Unrecognized organization '%!s!.'" on page 376
22	"Invalid certificate chain length (%!s!)." on page 375
23	"Certificate error (4023)." on page 379
24	"Server certificate not trusted." on page 379
25	"Unable to duplicate security context." on page 381
26	"Unable to attach the network layer to the security layer." on page 384
27	"Internal error 4027." on page 384
28	"Internal error 4028." on page 375
29	"Internal error 4029." on page 375
30	"Internal error 4030." on page 381
31	"Internal error 4031." on page 384
32	"Internal error 4032." on page 383
33	"Unable to open certificate file '%!s!.'" on page 378
34	"Unable to read certificates." on page 382
35	"Unable to read the private key." on page 383
36	"Unable to set the private key." on page 384
37	"Unable to fetch a certificate expiry date." on page 378
38	"Unable to copy a certificate." on page 381
39	"Unable to add a certificate to a certificate chain." on page 374
40	"Unable to find the trusted certificate file '%!s!.'" on page 386
41	"Error reading from the trusted certificate file '%!s!.'" on page 387
42	"No trusted certificates found." on page 377
43	"Unable to allocate a certificate." on page 380
44	"Unable to import a certificate." on page 382
45	"Internal initialization error 4045." on page 385

Error code	Error message
46	"Internal initialization error 4046." on page 385
47	"Unable to set the protocol side (%!s!)." on page 385
48	"Unable to add a trusted certificate." on page 374
49	"Unable to create a private key object." on page 380
50	"A certificate has expired." on page 378
51	"Unrecognized organization unit '%!s!.'" on page 377
52	"Unrecognized common name '%!s!.'" on page 376
53	"Handshake error." on page 382
54	"Unsupported HTTP version: %!s!" on page 368
55	"Internal initialization error 4055." on page 386
56	"Internal initialization error 4056." on page 386
57	"The host name '%!s!' could not be found." on page 392
58	"Unable to create a TCP/IP socket." on page 390
59	"Unable to create a UDP socket." on page 390
60	"Unable to bind a socket to port %!s!" on page 388
61	"Unable to clean up the socket layer." on page 389
62	"Unable to close a socket." on page 389
63	"Unable to connect a socket." on page 389
64	"Unable to get a socket's local name." on page 391
65	"Unable to get socket option number %!s!" on page 392
66	"Unable to set socket option number %!s!" on page 395
67	"Unable to listen on a socket. The backlog is %!s!" on page 393
68	"Unable to shut down a socket." on page 395
69	"Unable to select a socket status." on page 394
70	"Unable to initialize the sockets layer." on page 396
71	"Unable to determine localhost." on page 394
72	"Unable to get host by address." on page 391

Error code	Error message
73	"Unable to load the network interface library." on page 369
74	"Invalid port number %1s!. The value must be between zero and 65535." on page 394
75	"ActiveSync synchronization cannot be initiated by an application." on page 362
76	"ActiveSync provider has not been installed." on page 362
77	"The content type '%1s!' is unknown." on page 366
78	"Client id is not available for use in HTTP header." on page 365
79	"The HTTP buffer size specified is out of the valid range." on page 364
80	"Extra data found in the HTTP body: %1s!" on page 367
81	"Failed to read encoded CR LF." on page 366
82	"Failed to read CR LF." on page 366
83	"Timed out while waiting for the next HTTP request in this synchronization." on page 367
84	"Failed to read encoded chunk length." on page 365
85	"An unexpected character was read while parsing the chunk length. %1s!" on page 365
86	"An error status was returned: '%1s!'." on page 364
87	"Unknown transfer encoding: '%1s!'." on page 368
88	"Unable to parse cookie: '%1s!'." on page 368
89	"Expected data from remote but current request is not a POST." on page 367
200	"Invalid liveness timeout value %1s!. The value must be between zero and 65535." on page 393
201	"Timed out trying to read %1s! bytes." on page 374
202	"Timed out trying to write %1s! bytes." on page 397

Communication error messages sorted by message

Error code	Error message
50	"A certificate has expired." on page 378
76	"ActiveSync provider has not been installed." on page 362
75	"ActiveSync synchronization cannot be initiated by an application." on page 362
11	"An end read failed." on page 363
10	"An end write failed." on page 363
18	"An error occurred during shutdown." on page 388
86	"An error status was returned: '%!s!.'" on page 364
85	"An unexpected character was read while parsing the chunk length. %!s!." on page 365
23	"Certificate error (4023)." on page 379
78	"Client id is not available for use in HTTP header." on page 365
41	"Error reading from the trusted certificate file '%!s!.'" on page 387
89	"Expected data from remote but current request is not a POST." on page 367
80	"Extra data found in the HTTP body: %!s!" on page 367
82	"Failed to read CR LF." on page 366
81	"Failed to read encoded CR LF." on page 366
84	"Failed to read encoded chunk length." on page 365
12	"Feature not implemented." on page 370
53	"Handshake error." on page 382
27	"Internal error 4027." on page 384
28	"Internal error 4028." on page 375
29	"Internal error 4029." on page 375
30	"Internal error 4030." on page 381

Error code	Error message
31	"Internal error 4031." on page 384
32	"Internal error 4032." on page 383
45	"Internal initialization error 4045." on page 385
46	"Internal initialization error 4046." on page 385
55	"Internal initialization error 4055." on page 386
56	"Internal initialization error 4056." on page 386
22	"Invalid certificate chain length (%1s!)." on page 375
200	"Invalid liveness timeout value %1s!. The value must be between zero and 65535." on page 393
1	"Invalid parameter '%1s!.'" on page 370
74	"Invalid port number %1s!. The value must be between zero and 65535." on page 394
20	"Invalid root certificate." on page 380
0	"No error or unknown error." on page 370
42	"No trusted certificates found." on page 377
4	"Parameter value '%1s!' is not a valid boolean value. The value must be 0 or 1." on page 371
5	"Parameter value '%1s!' is not a valid hexadecimal value." on page 371
3	"Parameter value '%1s!' is not an unsigned integer value or range. A range has the form NNN-NNN." on page 372
2	"Parameter value '%1s!' is not an unsigned integer." on page 371
24	"Server certificate not trusted." on page 379
79	"The HTTP buffer size specified is out of the valid range." on page 364
77	"The content type '%1s!' is unknown." on page 366
57	"The host name '%1s!' could not be found." on page 392
13	"The operation would cause blocking." on page 396
201	"Timed out trying to read %1s! bytes." on page 374

Error code	Error message
202	“Timed out trying to write %1s! bytes.” on page 397
83	“Timed out while waiting for the next HTTP request in this synchronization.” on page 367
39	“Unable to add a certificate to a certificate chain.” on page 374
48	“Unable to add a trusted certificate.” on page 374
6	“Unable to allocate %1s! bytes.” on page 369
43	“Unable to allocate a certificate.” on page 380
26	“Unable to attach the network layer to the security layer.” on page 384
60	“Unable to bind a socket to port %1s!” on page 388
61	“Unable to clean up the socket layer.” on page 389
62	“Unable to close a socket.” on page 389
63	“Unable to connect a socket.” on page 389
38	“Unable to copy a certificate.” on page 381
58	“Unable to create a TCP/IP socket.” on page 390
59	“Unable to create a UDP socket.” on page 390
49	“Unable to create a private key object.” on page 380
17	“Unable to create a random number object.” on page 362
19	“Unable to dequeue from the connection queue.” on page 363
71	“Unable to determine localhost.” on page 394
25	“Unable to duplicate security context.” on page 381
37	“Unable to fetch a certificate expiry date.” on page 378
40	“Unable to find the trusted certificate file ‘%1s!’.” on page 386
14	“Unable to generate a random number.” on page 364
64	“Unable to get a socket’s local name.” on page 391
72	“Unable to get host by address.” on page 391

Error code	Error message
65	“Unable to get socket option number %1s!” on page 392
44	“Unable to import a certificate.” on page 382
15	“Unable to initialize the random number generator.” on page 369
70	“Unable to initialize the sockets layer.” on page 396
67	“Unable to listen on a socket. The backlog is %1s!” on page 393
73	“Unable to load the network interface library.” on page 369
33	“Unable to open certificate file ‘%1s!’.” on page 378
88	“Unable to parse cookie: ‘%1s!’.” on page 368
7	“Unable to parse the parameter string ‘%1s!’.” on page 372
8	“Unable to read %1s! bytes.” on page 373
34	“Unable to read certificates.” on page 382
35	“Unable to read the private key.” on page 383
16	“Unable to seed the random number generator.” on page 387
69	“Unable to select a socket status.” on page 394
66	“Unable to set socket option number %1s!” on page 395
36	“Unable to set the private key.” on page 384
47	“Unable to set the protocol side (%1s!).” on page 385
68	“Unable to shut down a socket.” on page 395
9	“Unable to write %1s! bytes.” on page 396
87	“Unknown transfer encoding: ‘%1s!’.” on page 368
52	“Unrecognized common name ‘%1s!’.” on page 376
21	“Unrecognized organization ‘%1s!’.” on page 376
51	“Unrecognized organization unit ‘%1s!’.” on page 377
54	“Unsupported HTTP version: %1s!” on page 368

Communication error messages sorted by constant

Constant	Error message
ACTSYNC NOT INSTALLED	“ActiveSync provider has not been installed.” on page 362
ACTSYNC NO PORT	“ActiveSync synchronization cannot be initiated by an application.” on page 362
CREATE RANDOM OBJECT	“Unable to create a random number object.” on page 362
DEQUEUEING CONNECTION	“Unable to dequeue from the connection queue.” on page 363
END READ	“An end read failed.” on page 363
END WRITE	“An end write failed.” on page 363
GENERATE RANDOM	“Unable to generate a random number.” on page 364
HTTP BAD STATUS CODE	“An error status was returned: ‘%1!s!’.” on page 364
HTTP BUFFER SIZE OUT OF RANGE	“The HTTP buffer size specified is out of the valid range.” on page 364
HTTP CHUNK LEN BAD CHARACTER	“An unexpected character was read while parsing the chunk length. %1!s!’.” on page 365
HTTP CHUNK LEN ENCODED MISSING	“Failed to read encoded chunk length.” on page 365
HTTP CLIENT ID NOT SET	“Client id is not available for use in HTTP header.” on page 365
HTTP CONTENT TYPE NOT SPECIFIED	“The content type ‘%1!s!’ is unknown.” on page 366
HTTP CRLF ENCODED MISSING	“Failed to read encoded CR LF.” on page 366
HTTP CRLF MISSING	“Failed to read CR LF.” on page 366
HTTP EXPECTED POST	“Expected data from remote but current request is not a POST.” on page 367

Constant	Error message
HTTP EXTRA DATA END READ	“Extra data found in the HTTP body: %1!s!” on page 367
HTTP NO CONTD CONNEC- TION	“Timed out while waiting for the next HTTP request in this synchronization.” on page 367
HTTP UNABLE TO PARSE COOKIE	“Unable to parse cookie: ‘%1!s!’.” on page 368
HTTP UNKNOWN TRANSFER ENCODING	“Unknown transfer encoding: ‘%1!s!’.” on page 368
HTTP VERSION	“Unsupported HTTP version: %1!s!” on page 368
INIT RANDOM	“Unable to initialize the random number generator.” on page 369
LOAD NETWORK LIBRARY	“Unable to load the network interface library.” on page 369
MEMORY ALLOCATION	“Unable to allocate %1!s! bytes.” on page 369
NONE	“No error or unknown error.” on page 370
NOT IMPLEMENTED	“Feature not implemented.” on page 370
PARAMETER	“Invalid parameter ‘%1!s!’.” on page 370
PARAMETER NOT BOOLEAN	“Parameter value ‘%1!s!’ is not a valid boolean value. The value must be 0 or 1.” on page 371
PARAMETER NOT HEX	“Parameter value ‘%1!s!’ is not a valid hexadecimal value.” on page 371
PARAMETER NOT UINT32	“Parameter value ‘%1!s!’ is not an unsigned integer.” on page 371
PARAMETER NOT UINT32 RANGE	“Parameter value ‘%1!s!’ is not an unsigned integer value or range. A range has the form NNN-NNN.” on page 372
PARSE	“Unable to parse the parameter string ‘%1!s!’.” on page 372

Constant	Error message
READ	“Unable to read %1s! bytes.” on page 373
READ TIMEOUT	“Timed out trying to read %1s! bytes.” on page 374
SECURE ADD CERTIFICATE	“Unable to add a certificate to a certificate chain.” on page 374
SECURE ADD TRUSTED CERTIFICATE	“Unable to add a trusted certificate.” on page 374
SECURE CERTIFICATE CHAIN FUNC	“Internal error 4028.” on page 375
SECURE CERTIFICATE CHAIN LENGTH	“Invalid certificate chain length (%1s!).” on page 375
SECURE CERTIFICATE CHAIN REF	“Internal error 4029.” on page 375
SECURE CERTIFICATE COMMON NAME	“Unrecognized common name ‘%1s!’.” on page 376
SECURE CERTIFICATE COMPANY NAME	“Unrecognized organization ‘%1s!’.” on page 376
SECURE CERTIFICATE COMPANY UNIT	“Unrecognized organization unit ‘%1s!’.” on page 377
SECURE CERTIFICATE COUNT	“No trusted certificates found.” on page 377
SECURE CERTIFICATE EXPIRED	“A certificate has expired.” on page 378
SECURE CERTIFICATE EXPIRY DATE	“Unable to fetch a certificate expiry date.” on page 378
SECURE CERTIFICATE FILE NOT FOUND	“Unable to open certificate file ‘%1s!’.” on page 378
SECURE CERTIFICATE NOT TRUSTED	“Server certificate not trusted.” on page 379
SECURE CERTIFICATE REF	“Certificate error (4023).” on page 379
SECURE CERTIFICATE ROOT	“Invalid root certificate.” on page 380

Constant	Error message
SECURE CREATE CERTIFICATE	“Unable to allocate a certificate.” on page 380
SECURE CREATE PRIVATE KEY OBJECT	“Unable to create a private key object.” on page 380
SECURE DUPLICATE CONTEXT	“Unable to duplicate security context.” on page 381
SECURE ENABLE NON BLOCKING	“Internal error 4030.” on page 381
SECURE EXPORT CERTIFICATE	“Unable to copy a certificate.” on page 381
SECURE HANDSHAKE	“Handshake error.” on page 382
SECURE IMPORT CERTIFICATE	“Unable to import a certificate.” on page 382
SECURE READ CERTIFICATE	“Unable to read certificates.” on page 382
SECURE READ PRIVATE KEY	“Unable to read the private key.” on page 383
SECURE SET CHAIN NUMBER	“Internal error 4032.” on page 383
SECURE SET CIPHER SUITES	“Internal error 4031.” on page 384
SECURE SET IO	“Unable to attach the network layer to the security layer.” on page 384
SECURE SET IO SEMANTICS	“Internal error 4027.” on page 384
SECURE SET PRIVATE KEY	“Unable to set the private key.” on page 384
SECURE SET PROTOCOL SIDE	“Unable to set the protocol side (%!s!).” on page 385
SECURE SET RANDOM FUNC	“Internal initialization error 4046.” on page 385
SECURE SET RANDOM REF	“Internal initialization error 4045.” on page 385
SECURE SET READ FUNC	“Internal initialization error 4055.” on page 386

Constant	Error message
SECURE SET WRITE FUNC	“Internal initialization error 4056.” on page 386
SECURE TRUSTED CERTIFICATE FILE NOT FOUND	“Unable to find the trusted certificate file ‘%1s!’.” on page 386
SECURE TRUSTED CERTIFICATE READ	“Error reading from the trusted certificate file ‘%1s!’.” on page 387
SEED RANDOM	“Unable to seed the random number generator.” on page 387
SHUTTING DOWN	“An error occurred during shutdown.” on page 388
SOCKET BIND	“Unable to bind a socket to port %1s!” on page 388
SOCKET CLEANUP	“Unable to clean up the socket layer.” on page 389
SOCKET CLOSE	“Unable to close a socket.” on page 389
SOCKET CONNECT	“Unable to connect a socket.” on page 389
SOCKET CREATE TCPIP	“Unable to create a TCP/IP socket.” on page 390
SOCKET CREATE UDP	“Unable to create a UDP socket.” on page 390
SOCKET GET HOST BY ADDR	“Unable to get host by address.” on page 391
SOCKET GET NAME	“Unable to get a socket’s local name.” on page 391
SOCKET GET OPTION	“Unable to get socket option number %1s!” on page 392
SOCKET HOST NAME NOT FOUND	“The host name ‘%1s!’ could not be found.” on page 392
SOCKET LISTEN	“Unable to listen on a socket. The backlog is %1s!” on page 393
SOCKET LIVENESS OUT OF RANGE	“Invalid liveness timeout value %1s!. The value must be between zero and 65535.” on page 393

Constant	Error message
SOCKET LOCALHOST NAME NOT FOUND	“Unable to determine localhost.” on page 394
SOCKET PORT OUT OF RANGE	“Invalid port number %1!s!. The value must be between zero and 65535.” on page 394
SOCKET SELECT	“Unable to select a socket status.” on page 394
SOCKET SET OPTION	“Unable to set socket option number %1!s!” on page 395
SOCKET SHUTDOWN	“Unable to shut down a socket.” on page 395
SOCKET STARTUP	“Unable to initialize the sockets layer.” on page 396
WOULD BLOCK	“The operation would cause blocking.” on page 396
WRITE	“Unable to write %1!s! bytes.” on page 396
WRITE TIMEOUT	“Timed out trying to write %1!s! bytes.” on page 397

Communication error descriptions

This section provides a full listing of error messages and descriptions.

Errors with an ODBC state marked “handled by ODBC driver” are not returned to ODBC applications, as the ODBC driver carries out the required actions.

ActiveSync provider has not been installed.

Item	Value
Error code	76
Constant	ACTSYNC_NOT_INSTALLED (Java) STREAM_ERROR_ACTSYNC_NOT_INSTALLED (C/C++) ulStreamErrorActsSyncNotInstalled (Visual Basic)

Probable cause The ActiveSync provider has not been installed. Run dbasinst to install it (see documentation for details).

ActiveSync synchronization cannot be initiated by an application.

Item	Value
Error code	75
Constant	ACTSYNC_NO_PORT (Java) STREAM_ERROR_ACTSYNC_NO_PORT (C/C++) ulStreamErrorActsSyncNoPort (Visual Basic)

Probable cause ActiveSync synchronization can only be initiated by ActiveSync itself, either by placing the device in its cradle or by selecting “Synchronize” from the ActiveSync Manager. To initiate a synchronization from an application, use the TCP/IP socket synchronization stream.

Unable to create a random number object.

Item	Value
Error code	17
Constant	CREATE_RANDOM_OBJECT (Java) STREAM_ERROR_CREATE_RANDOM_OBJECT (C/C++) ulStreamErrorCreateRandomObject (Visual Basic)

Probable cause The secure network layer could not create a random-number-generating

object. Free up system resources, reconnect and retry the operation.

Unable to dequeue from the connection queue.

Item	Value
Error code	19
Constant	DEQUEUEING_CONNECTION (Java) STREAM_ERROR_DEQUEUEING_CONNECTION (C/C++) ulStreamErrorDequeueingConnection (Visual Basic)

Probable cause The MobiLink synchronization server encountered an error while attempting to get a queued connection (synchronization) request. Free up system resources. If the problem persists, restart the MobiLink synchronization server.

An end read failed.

Item	Value
Error code	11
Constant	END_READ (Java) STREAM_ERROR_END_READ (C/C++) ulStreamErrorEndRead (Visual Basic)

Probable cause Unable to finish a sequence of reads from the network.
See also: READ

An end write failed.

Item	Value
Error code	10
Constant	END_WRITE (Java) STREAM_ERROR_END_WRITE (C/C++) ulStreamErrorEndWrite (Visual Basic)

Probable cause Unable to finish a sequence of writes to the network.
See also: WRITE

Unable to generate a random number.

Item	Value
Error code	14
Constant	GENERATE_RANDOM (Java) STREAM_ERROR_- GENERATE_RANDOM (C/C++) ulStreamErrorGener- ateRandom (Visual Basic)

Probable cause The secure network layer requires a random number but was unable to generate one. Free up system resources, reconnect and retry the operation.

An error status was returned: '%1!s!'.

Item	Value
Error code	86
Constant	HTTP_BAD_STATUS_CODE (Java) STREAM_- ERROR_HTTP_BAD_STATUS_CODE (C/C++) ul- StreamErrorHttpBadStatusCode (Visual Basic)
Parameter 1	The status line read.

Probable cause Examine the status line to determine the cause of the failure.

The HTTP buffer size specified is out of the valid range.

Item	Value
Error code	79
Constant	HTTP_BUFFER_SIZE_OUT_OF_RANGE (Java) STREAM_ERROR_HTTP_BUFFER_SIZE_OUT_OF_- RANGE (C/C++) ulStreamErrorHttpBufferSizeOut- OfRange (Visual Basic)

Probable cause Fix the HTTP buffer size. A valid buffer size is positive and not overly large for the host platform.

**An unexpected character was read while parsing the chunk length.
%1!s!.**

Item	Value
Error code	85
Constant	HTTP_CHUNK_LEN_BAD_CHARACTER (Java) STREAM_ERROR_HTTP_CHUNK_LEN_BAD_- CHARACTER (C/C++) ulStreamErrorHttpChunkLen- BadCharacter (Visual Basic)
Parameter 1	The unexpected character.

Probable cause Try using a fixed length HTTP body.

Failed to read encoded chunk length.

Item	Value
Error code	84
Constant	HTTP_CHUNK_LEN_ENCODED_MISSING (Java) STREAM_ERROR_HTTP_CHUNK_LEN_ENCODED_- MISSING (C/C++) ulStreamErrorHttpChunkLenEncoded- Missing (Visual Basic)

Probable cause Try using a fixed length HTTP body.

Client id is not available for use in HTTP header.

Item	Value
Error code	78
Constant	HTTP_CLIENT_ID_NOT_SET (Java) STREAM_- ERROR_HTTP_CLIENT_ID_NOT_SET (C/C++) ul- StreamErrorHttpClientIdNotSet (Visual Basic)

Probable cause The client id was not passed into the HTTP client code. Contact technical support for a fix.

The content type ‘%1!s!’ is unknown.

Item	Value
Error code	77
Constant	HTTP_CONTENT_TYPE_NOT_SPECIFIED (Java) STREAM_ERROR_HTTP_CONTENT_TYPE_NOT_- SPECIFIED (C/C++) ulStreamErrorHttpContentTypeNot- Specified (Visual Basic)
Parameter 1	The content type.

Probable cause An unknown content type was specified. Refer to the documentation and change the content type to one of the supported types.

Failed to read encoded CR LF.

Item	Value
Error code	81
Constant	HTTP_CRLF_ENCODED_MISSING (Java) STREAM_- ERROR_HTTP_CRLF_ENCODED_MISSING (C/C++) ulStreamErrorHttpCrlfEncodedMissing (Visual Basic)

Probable cause The proxy you are using may not be compatible with MobiLink. Please check your configuration.

Failed to read CR LF.

Item	Value
Error code	82
Constant	HTTP_CRLF_MISSING (Java) STREAM_ERROR_- HTTP_CRLF_MISSING (C/C++) ulStreamEr- rorHttpCrlfMissing (Visual Basic)

Probable cause The proxy you are using may not be compatible with MobiLink. Please check your configuration.

Expected data from remote but current request is not a POST.

Item	Value
Error code	89
Constant	HTTP_EXPECTED_POST (Java) STREAM_ERROR_HTTP_EXPECTED_POST (C/C++) ulStreamErrorHttpExpectedPost (Visual Basic)

Probable cause The proxy you are using may not be compatible with MobiLink. Please check your configuration.

Extra data found in the HTTP body: %!s!

Item	Value
Error code	80
Constant	HTTP_EXTRA_DATA_END_READ (Java) STREAM_ERROR_HTTP_EXTRA_DATA_END_READ (C/C++) ulStreamErrorHttpExtraDataEndRead (Visual Basic)
Parameter 1	First few characters in the extra data.

Probable cause Extra data has been introduced into the HTTP body. This may have been added by a proxy agent. Try eliminating the proxy.

Timed out while waiting for the next HTTP request in this synchronization.

Item	Value
Error code	83
Constant	HTTP_NO_CONTD_CONNECTION (Java) STREAM_ERROR_HTTP_NO_CONTD_CONNECTION (C/C++) ulStreamErrorHttpNoContdConnection (Visual Basic)

Probable cause The server timed out while waiting for the next HTTP request from the remote site. Determine why this request failed to reach the server or try a persistent connection.

Unable to parse cookie: '%1!s!'.

Item	Value
Error code	88
Constant	HTTP_UNABLE_TO_PARSE_COOKIE (Java) STREAM_ERROR_HTTP_UNABLE_TO_PARSE_COOKIE (C/C++) ulStreamErrorHttpUnableToParseCookie (Visual Basic)
Parameter 1	The set cookie header.

Probable cause Determine where the set cookie header is being corrupted.

Unknown transfer encoding: '%1!s!'.

Item	Value
Error code	87
Constant	HTTP_UNKNOWN_TRANSFER_ENCODING (Java) STREAM_ERROR_HTTP_UNKNOWN_TRANSFER_ENCODING (C/C++) ulStreamErrorHttpUnknownTransferEncoding (Visual Basic)
Parameter 1	The unknown encoding.

Probable cause Determine how the unknown transfer encoding is getting generated.

Unsupported HTTP version: %1!s!

Item	Value
Error code	54
Constant	HTTP_VERSION (Java) STREAM_ERROR_HTTP_VERSION (C/C++) ulStreamErrorHttpVersion (Visual Basic)
Parameter 1	The requested HTTP version.

Probable cause The requested HTTP version is unsupported. Consult the documentation and specify a supported HTTP version. At the time of publication the supported HTTP versions are 1.0 and 1.1.

Unable to initialize the random number generator.

Item	Value
Error code	15
Constant	INIT_RANDOM (Java) STREAM_ERROR_INIT_RANDOM (C/C++) ulStreamErrorInitRandom (Visual Basic)

Probable cause The secure network layer could not initialize its random number generator. Free up system resources, reconnect and retry the operation.

Unable to load the network interface library.

Item	Value
Error code	73
Constant	LOAD_NETWORK_LIBRARY (Java) STREAM_ERROR_LOAD_NETWORK_LIBRARY (C/C++) ulStreamErrorLoadNetworkLibrary (Visual Basic)

Probable cause The network interface library could not be found and/or loaded. Please check the following:

- 1) The sockets layer is properly installed. The correct network interface library (or DLL or shared object) must be present and accessible.
- 2) There are enough system resources available. Free up system resources if they are running low.

Unable to allocate %1s! bytes.

Item	Value
Error code	6
Constant	MEMORY_ALLOCATION (Java) STREAM_ERROR_MEMORY_ALLOCATION (C/C++) ulStreamErrorMemoryAllocation (Visual Basic)
Parameter 1	The number of bytes that was requested.

Probable cause The network layer was unable to allocate the given number of bytes of storage. Free up system memory and retry the operation. The technique used to free up system memory depends on the operating system and how it is

configured. The simplest technique is to reduce the number of active processes. Consult your operating system documentation for details.

No error or unknown error.

Item	Value
Error code	0
Constant	NONE (Java) STREAM_ERROR_NONE (C/C++) ulStreamErrorNone (Visual Basic)

Probable cause This code indicates there was either no network error, or an unknown network error occurred.

Feature not implemented.

Item	Value
Error code	12
Constant	NOT_IMPLEMENTED (Java) STREAM_ERROR_NOT_IMPLEMENTED (C/C++) ulStreamErrorNotImplemented (Visual Basic)

Probable cause An unimplemented internal feature was requested. Please contact technical support.

Invalid parameter '%1!s!'.

Item	Value
Error code	1
Constant	PARAMETER (Java) STREAM_ERROR_PARAMETER (C/C++) ulStreamErrorParameter (Visual Basic)
Parameter 1	The invalid parameter value.

Probable cause Network parameters are of the form “name=value;[name2=value2[;...]]”. This code indicates an invalid parameter value. Consult the documentation for the corresponding parameter name, and correct the parameter value.

Parameter value ‘%1!s!’ is not a valid boolean value. The value must be 0 or 1.

Item	Value
Error code	4
Constant	PARAMETER_NOT_BOOLEAN (Java) STREAM_ERROR_PARAMETER_NOT_BOOLEAN (C/C++) ulStreamErrorParameterNotBoolean (Visual Basic)
Parameter 1	The invalid parameter value.

Probable cause Network parameters are of the form “name=value;[name2=value2[;...]]”. The parameter value is not a boolean value. Locate the offending parameter specification and change the value of the parameter to either 0 (for off or false) or 1 (for on or true).

Parameter value ‘%1!s!’ is not a valid hexadecimal value.

Item	Value
Error code	5
Constant	PARAMETER_NOT_HEX (Java) STREAM_ERROR_PARAMETER_NOT_HEX (C/C++) ulStreamErrorParameterNotHex (Visual Basic)
Parameter 1	The invalid parameter value.

Probable cause Network parameters are of the form “name=value;[name2=value2[;...]]”. The parameter value is not a hexadecimal (base 16) value. Locate the offending parameter specification and change the value of the parameter to a hexadecimal value.

Parameter value ‘%1!s!’ is not an unsigned integer.

Item	Value
Error code	2
Constant	PARAMETER_NOT_UINT32 (Java) STREAM_ERROR_PARAMETER_NOT_UINT32 (C/C++) ulStreamErrorParameterNotUInt32 (Visual Basic)
Parameter 1	The invalid parameter value.

Probable cause Network parameters are of the form “name=value;[name2=value2[;...]]”. The parameter value is not an unsigned integer. Locate the offending parameter specification and change the value of the parameter to an unsigned integer.

Parameter value ‘%1!s!’ is not an unsigned integer value or range. A range has the form NNN-NNN.

Item	Value
Error code	3
Constant	PARAMETER_NOT_UINT32_RANGE (Java) STREAM_ERROR_PARAMETER_NOT_UINT32_- RANGE (C/C++) ulStreamErrorParameterNotU- int32Range (Visual Basic)
Parameter 1	The invalid parameter value.

Probable cause Network parameters are of the form “name=value;[name2=value2[;...]]”. The parameter value is not an unsigned integer value or range. Locate the offending parameter specification and change the value of the parameter to an unsigned integer or an unsigned range. An unsigned range has the form: NNN-NNN.

Unable to parse the parameter string ‘%1!s!’.

Item	Value
Error code	7
Constant	PARSE (Java) STREAM_ERROR_PARSE (C/C++) ul- StreamErrorParse (Visual Basic)
Parameter 1	The parameter string that could not be parsed.

Probable cause Network parameters are of the form “name=value;[name2=value2[;...]]”. Optionally, the entire list of parameters may be enclosed in parentheses. The given string does not follow this convention. Inspect the string, fix any formatting problems, and retry the operation.

Unable to read %1!s! bytes.

Item	Value
Error code	8
Constant	READ (Java) STREAM_ERROR_READ (C/C++) ul-StreamErrorRead (Visual Basic)
Parameter 1	The number of bytes that could not be read.

Probable cause

Unable to read the given number of bytes from the network layer. Note that reads may occur as part of any larger network operation. For example, some network layers have sub-layers that perform several reads and writes as part of a basic operation in the upper layer.

The cause of a read error is usually one of the following:

1) The network had a problem that caused the read to fail.

Reconnect and retry the operation.

2) The connection timed out.

Reconnect and retry the operation.

3) The other side of the connection cleanly terminated the connection.

Consult the client and/or server logs for errors that indicate why the connection has been dropped.

Consult the output-log errors and fix the cause, then retry the operation.

4) The process at the other side of the connection was aborted.

Consult the client and/or server output logs for errors that indicate why the process was aborted.

If the process was shut down by other than normal means, there may not be any errors in its output log.

Reconnect and retry the operation.

5) The system is low on resources, and cannot perform the read.

Free up system resources, reconnect and retry the operation. If subsequent retry attempts fail, consult your network administrator.

Timed out trying to read %1!s! bytes.

Item	Value
Error code	201
Constant	READ_TIMEOUT (Java) STREAM_ERROR_READ_TIMEOUT (C/C++) ulStreamErrorReadTimeout (Visual Basic)
Parameter 1	The number of bytes that could not be read.

Probable cause Unable to read the given number of bytes from the network layer in the given time.

Check that the network is functioning correctly, and that the sending application is still running.

Unable to add a certificate to a certificate chain.

Item	Value
Error code	39
Constant	SECURE_ADD_CERTIFICATE (Java) STREAM_ERROR_SECURE_ADD_CERTIFICATE (C/C++) ulStreamErrorSecureAddCertificate (Visual Basic)

Probable cause The secure network layer was unable to add a certificate to a certificate chain. Free up system resources and retry the operation.

Unable to add a trusted certificate.

Item	Value
Error code	48
Constant	SECURE_ADD_TRUSTED_CERTIFICATE (Java) STREAM_ERROR_SECURE_ADD_TRUSTED_CERTIFICATE (C/C++) ulStreamErrorSecureAddTrustedCertificate (Visual Basic)

Probable cause The secure network layer was unable to add a trusted certificate to a certificate chain. The most likely cause is a shortage of system resources. Free up system resources and retry the operation.

Internal error 4028.

Item	Value
Error code	28
Constant	SECURE_CERTIFICATE_CHAIN_FUNC (Java) STREAM_ERROR_SECURE_CERTIFICATE_CHAIN_- FUNC (C/C++) ulStreamErrorSecureCertificateChainFunc (Visual Basic)

Probable cause An internal error has occurred in the network layer. Please contact technical support.

Invalid certificate chain length (%1!s!).

Item	Value
Error code	22
Constant	SECURE_CERTIFICATE_CHAIN_LENGTH (Java) STREAM_ERROR_SECURE_CERTIFICATE_CHAIN_- LENGTH (C/C++) ulStreamErrorSecureCertificateChain- Length (Visual Basic)
Parameter 1	The certificate chain length.

Probable cause The certificate chain has the wrong length. This is an internal error that should never occur. Please contact technical support.

Internal error 4029.

Item	Value
Error code	29
Constant	SECURE_CERTIFICATE_CHAIN_REF (Java) STREAM_ERROR_SECURE_CERTIFICATE_CHAIN_- REF (C/C++) ulStreamErrorSecureCertificateChainRef (Visual Basic)

Probable cause An internal error has occurred in the network layer. Please contact technical support.

Unrecognized common name ‘%1!s!’.

Item	Value
Error code	52
Constant	SECURE_CERTIFICATE_COMMON_NAME (Java) STREAM_ERROR_SECURE_CERTIFICATE_- COMMON_NAME (C/C++) ulStreamErrorSecureCer- tificateCommonName (Visual Basic)
Parameter 1	The common name.

Probable cause The given common name is not in the certificate chain. Check the following:

- 1) The common name was properly entered.
- 2) The correct certificate file was specified.
- 3) The common name is in the certificate chain. You can verify this with the readcert utility.

Unrecognized organization ‘%1!s!’.

Item	Value
Error code	21
Constant	SECURE_CERTIFICATE_COMPANY_NAME (Java) STREAM_ERROR_SECURE_CERTIFICATE_- COMPANY_NAME (C/C++) ulStreamErrorSecureCer- tificateCompanyName (Visual Basic)
Parameter 1	The organization name.

Probable cause The given organization name is not in the certificate chain. Check the following:

- 1) The organization name was properly entered.
- 2) The correct certificate file was specified.
- 3) The organization name is in the certificate chain. You can verify this with the readcert utility.

Unrecognized organization unit '%1!s!'.

Item	Value
Error code	51
Constant	SECURE_CERTIFICATE_COMPANY_UNIT (Java) STREAM_ERROR_SECURE_CERTIFICATE_- COMPANY_UNIT (C/C++) ulStreamErrorSecureCer- tificateCompanyUnit (Visual Basic)
Parameter 1	The organization unit name.

Probable cause

The given organization unit is not in the certificate chain. Check the following:

- 1) The in company name was properly entered.
- 2) The correct certificate file was specified.
- 3) The company name is in the certificate chain. You can verify this with the readcert utility.

No trusted certificates found.

Item	Value
Error code	42
Constant	SECURE_CERTIFICATE_COUNT (Java) STREAM_- ERROR_SECURE_CERTIFICATE_COUNT (C/C++) ulStreamErrorSecureCertificateCount (Visual Basic)

Probable cause

The given file does not contain a certificate. Check the following:

- 1) The certificate file name was properly specified.
- 2) The certificate file contains one or more certificates.
- 3) The certificate file contains the correct certificate(s).

A certificate has expired.

Item	Value
Error code	50
Constant	SECURE_CERTIFICATE_EXPIRED (Java) STREAM_ERROR_SECURE_CERTIFICATE_EXPIRED (C/C++) ulStreamErrorSecureCertificateExpired (Visual Basic)

Probable cause A certificate in the certificate chain has expired. Obtain a new certificate with a later expiry date and retry the operation.

Unable to fetch a certificate expiry date.

Item	Value
Error code	37
Constant	SECURE_CERTIFICATE_EXPIRY_DATE (Java) STREAM_ERROR_SECURE_CERTIFICATE_EXPIRY_DATE (C/C++) ulStreamErrorSecureCertificateExpiryDate (Visual Basic)

Probable cause A certificate's expiry date could not be read. Check the following:

- 1) The password was entered correctly.
- 2) The certificate file contains one or more certificates.
- 3) The certificate file contains the correct certificate(s).
- 4) The certificate file is undamaged.

Unable to open certificate file '%1!s!'.

Item	Value
Error code	33
Constant	SECURE_CERTIFICATE_FILE_NOT_FOUND (Java) STREAM_ERROR_SECURE_CERTIFICATE_FILE_NOT_FOUND (C/C++) ulStreamErrorSecureCertificateFileNotFound (Visual Basic)
Parameter 1	The certificate file name.

Probable cause The certificate file could not be opened. Check the following:

- 1) The certificate file name was properly specified.
- 2) The certificate file exists.
- 3) The certificate file contains one or more certificates.
- 4) The certificate file contains the correct certificate(s).
- 5) The program attempting to open the certificate file has sufficient privileges to read the file. This only applies to operating systems having user and/or file permissions.

Server certificate not trusted.

Item	Value
Error code	24
Constant	SECURE_CERTIFICATE_NOT_TRUSTED (Java) STREAM_ERROR_SECURE_CERTIFICATE_NOT_TRUSTED (C/C++) ulStreamErrorSecureCertificateNotTrusted (Visual Basic)

Probable cause The server's certificate was not signed by a trusted authority. Check the following:

- 1) The certificate file name was properly specified.
- 2) The certificate file contains one or more certificates.
- 3) The certificate file contains the correct certificate(s).
- 4) The client's list of trusted root certificates includes the server's root certificate.

Certificate error (4023).

Item	Value
Error code	23
Constant	SECURE_CERTIFICATE_REF (Java) STREAM_ERROR_SECURE_CERTIFICATE_REF (C/C++) ulStreamErrorSecureCertificateRef (Visual Basic)

Probable cause This is an internal error in the secure network layer. This is an internal error that should never occur. Please contact technical support.

Invalid root certificate.

Item	Value
Error code	20
Constant	SECURE_CERTIFICATE_ROOT (Java) STREAM_ERROR_SECURE_CERTIFICATE_ROOT (C/C++) ulStreamErrorSecureCertificateRoot (Visual Basic)

Probable cause The root certificate in the chain is invalid. At the time of publication, this error was defined but not used.

Unable to allocate a certificate.

Item	Value
Error code	43
Constant	SECURE_CREATE_CERTIFICATE (Java) STREAM_ERROR_SECURE_CREATE_CERTIFICATE (C/C++) ulStreamErrorSecureCreateCertificate (Visual Basic)

Probable cause The secure network layer was unable to allocate storage for a certificate. Free up system resources and retry the operation.

Unable to create a private key object.

Item	Value
Error code	49
Constant	SECURE_CREATE_PRIVATE_KEY_OBJECT (Java) STREAM_ERROR_SECURE_CREATE_PRIVATE_KEY_OBJECT (C/C++) ulStreamErrorSecureCreatePrivateKeyObject (Visual Basic)

Probable cause The secure network layer was unable to create a private key object, prior to loading the private key. The most likely cause is a shortage of system resources. Free up system resources and retry the operation.

Unable to duplicate security context.

Item	Value
Error code	25
Constant	SECURE_DUPLICATE_CONTEXT (Java) STREAM_ ERROR_SECURE_DUPLICATE_CONTEXT (C/C++) ulStreamErrorSecureDuplicateContext (Visual Basic)

Probable cause The secure network layer was unable to duplicate a security context.
Free up system resources and retry the operation.

Internal error 4030.

Item	Value
Error code	30
Constant	SECURE_ENABLE_NON_BLOCKING (Java) STREAM_ERROR_SECURE_ENABLE_NON_ BLOCKING (C/C++) ulStreamErrorSecureEnableNon- Blocking (Visual Basic)

Probable cause An internal error has occurred in the network layer. Please contact technical support.

Unable to copy a certificate.

Item	Value
Error code	38
Constant	SECURE_EXPORT_CERTIFICATE (Java) STREAM_ ERROR_SECURE_EXPORT_CERTIFICATE (C/C++) ulStreamErrorSecureExportCertificate (Visual Basic)

Probable cause The secure network layer was unable to copy a certificate. Free up system resources and retry the operation.

Handshake error.

Item	Value
Error code	53
Constant	SECURE_HANDSHAKE (Java) STREAM_ERROR_- SECURE_HANDSHAKE (C/C++) ulStreamErrorSecure- Handshake (Visual Basic)

Probable cause The secure handshake failed. Check the following:

- 1) On the client, the correct host machine and port number were specified.
- 2) On the server, the correct port number was specified.
- 3) The correct certificate file was specified, both on the client and on the server.

Unable to import a certificate.

Item	Value
Error code	44
Constant	SECURE_IMPORT_CERTIFICATE (Java) STREAM_- ERROR_SECURE_IMPORT_CERTIFICATE (C/C++) ulStreamErrorSecureImportCertificate (Visual Basic)

Probable cause The secure network layer was unable to import a certificate. Check the following:

- 1) The certificate file name was properly specified.
- 2) The certificate file exists.
- 3) The certificate file contains one or more certificates.
- 4) The certificate file contains the correct certificate(s).

Unable to read certificates.

Item	Value
Error code	34
Constant	SECURE_READ_CERTIFICATE (Java) STREAM_- ERROR_SECURE_READ_CERTIFICATE (C/C++) ul- StreamErrorSecureReadCertificate (Visual Basic)

Probable cause The certificate file could not be read. Check the following:

- 1) The password was entered correctly.
- 2) The certificate file contains one or more certificates.
- 3) The certificate file contains the correct certificate(s).
- 4) The certificate file is undamaged.

Unable to read the private key.

Item	Value
Error code	35
Constant	SECURE_READ_PRIVATE_KEY (Java) STREAM_ ERROR_SECURE_READ_PRIVATE_KEY (C/C++) ul- StreamErrorSecureReadPrivateKey (Visual Basic)

Probable cause The private key could not be read from the certificate file. Check the following:

- 1) The password was entered correctly.
- 2) The certificate file contains one or more certificates.
- 3) The certificate file contains the correct certificate(s).
- 4) The certificate file is undamaged.

Internal error 4032.

Item	Value
Error code	32
Constant	SECURE_SET_CHAIN_NUMBER (Java) STREAM_ ERROR_SECURE_SET_CHAIN_NUMBER (C/C++) ulStreamErrorSecureSetChainNumber (Visual Basic)

Probable cause An internal error has occurred in the network layer. Please contact technical support.

Internal error 4031.

Item	Value
Error code	31
Constant	SECURE_SET_CIPHER_SUITES (Java) STREAM_ERROR_SECURE_SET_CIPHER_SUITES (C/C++) ulStreamErrorSecureSetCipherSuites (Visual Basic)

Probable cause An internal error has occurred in the network layer. Please contact technical support.

Unable to attach the network layer to the security layer.

Item	Value
Error code	26
Constant	SECURE_SET_IO (Java) STREAM_ERROR_SECURE_SET_IO (C/C++) ulStreamErrorSecureSetIo (Visual Basic)

Probable cause The secure network layer was unable to attach to the network layer. Free up system resources and retry the operation.

Internal error 4027.

Item	Value
Error code	27
Constant	SECURE_SET_IO_SEMANTICS (Java) STREAM_ERROR_SECURE_SET_IO_SEMANTICS (C/C++) ulStreamErrorSecureSetIoSemantics (Visual Basic)

Probable cause An internal error has occurred in the network layer. Please contact technical support.

Unable to set the private key.

Item	Value
Error code	36
Constant	SECURE_SET_PRIVATE_KEY (Java) STREAM_ERROR_SECURE_SET_PRIVATE_KEY (C/C++) ulStreamErrorSecureSetPrivateKey (Visual Basic)

Probable cause The private key could not be used. Check the following:

- 1) The password was entered correctly.
- 2) The certificate file contains one or more certificates.
- 3) The certificate file contains the correct certificate(s).
- 4) The certificate file is undamaged.

Unable to set the protocol side (%1!s!).

Item	Value
Error code	47
Constant	SECURE_SET_PROTOCOL_SIDE (Java) STREAM_- ERROR_SECURE_SET_PROTOCOL_SIDE (C/C++) ulStreamErrorSecureSetProtocolSide (Visual Basic)
Parameter 1	The server side being set. The value is 1 for server side, and 2 for client side.

Probable cause The secure network layer was unable to establish the given protocol side.
This is an internal error that should never occur. Please contact technical
support.

Internal initialization error 4046.

Item	Value
Error code	46
Constant	SECURE_SET_RANDOM_FUNC (Java) STREAM_- ERROR_SECURE_SET_RANDOM_FUNC (C/C++) ul- StreamErrorSecureSetRandomFunc (Visual Basic)

Probable cause An internal error has occurred in the network layer. Please contact technical
support.

Internal initialization error 4045.

Item	Value
Error code	45
Constant	SECURE_SET_RANDOM_REF (Java) STREAM_- ERROR_SECURE_SET_RANDOM_REF (C/C++) ul- StreamErrorSecureSetRandomRef (Visual Basic)

Probable cause An internal error has occurred in the network layer. Please contact technical support.

Internal initialization error 4055.

Item	Value
Error code	55
Constant	SECURE_SET_READ_FUNC (Java) STREAM_ERROR_SECURE_SET_READ_FUNC (C/C++) ulStreamErrorSecureSetReadFunc (Visual Basic)

Probable cause This initialization error is most likely due to a lack of system resources. Free up system resources and retry the operation.

Internal initialization error 4056.

Item	Value
Error code	56
Constant	SECURE_SET_WRITE_FUNC (Java) STREAM_ERROR_SECURE_SET_WRITE_FUNC (C/C++) ulStreamErrorSecureSetWriteFunc (Visual Basic)

Probable cause This initialization error is most likely due to a lack of system resources. Free up system resources and retry the operation.

Unable to find the trusted certificate file '%1!s!'.

Item	Value
Error code	40
Constant	SECURE_TRUSTED_CERTIFICATE_FILE_NOT_FOUND (Java) STREAM_ERROR_SECURE_TRUSTED_CERTIFICATE_FILE_NOT_FOUND (C/C++) ulStreamErrorSecureTrustedCertificateFileNotFound (Visual Basic)
Parameter 1	The trusted certificate file name.

Probable cause The certificate file could not be found. Check the following:

- 1) The certificate file name was properly specified.
- 2) The certificate file exists.

- 3) The certificate file contains one or more certificates.
- 4) The certificate file contains the correct certificate(s).
- 5) The program attempting to open the certificate file has sufficient privileges to see the file. This only applies to operating systems having user and/or file permissions.

Error reading from the trusted certificate file '%!s!'.

Item	Value
Error code	41
Constant	SECURE_TRUSTED_CERTIFICATE_READ (Java) STREAM_ERROR_SECURE_TRUSTED_- CERTIFICATE_READ (C/C++) ulStreamErrorSe- cureTrustedCertificateRead (Visual Basic)
Parameter 1	The trusted certificate file name.

Probable cause The secure network layer was unable to read the trusted certificate file.
Check the following:

- 1) The certificate file name was properly specified.
- 2) The certificate file exists.
- 3) The certificate file contains one or more certificates.
- 4) The certificate file contains the correct certificate(s).
- 5) The program attempting to open the certificate file has sufficient privileges to see the file. This only applies to operating systems having user and/or file permissions.

Unable to seed the random number generator.

Item	Value
Error code	16
Constant	SEED_RANDOM (Java) STREAM_ERROR_SEED_- RANDOM (C/C++) ulStreamErrorSeedRandom (Visual Basic)

Probable cause The secure network layer could not seed its random number generator. Free
up system resources, reconnect and retry the operation.

An error occurred during shutdown.

Item	Value
Error code	18
Constant	SHUTTING_DOWN (Java) STREAM_ERROR_SHUTTING_DOWN (C/C++) ulStreamErrorShutting-Down (Visual Basic)

Probable cause The MobiLink synchronization server encountered an error in the network layer during shutdown. It is possible that some network operations pending at the time of shutdown were affected.

Unable to bind a socket to port %1!s!.

Item	Value
Error code	60
Constant	SOCKET_BIND (Java) STREAM_ERROR_SOCKET_BIND (C/C++) ulStreamErrorSocketBind (Visual Basic)
Parameter 1	The port number.

Probable cause The network layer was unable to bind a socket to the given port. Check the following.

1) (Server only) Verify that the port isn't already in use. If the port is in use, either shut down the application listening on that port, or specify a different port.

2) (Server only) Verify that there are no firewall restrictions on the use of the port.

3) (Client only) If the client_port option was used, verify that the given port isn't already in use. If only one client port was specified, consider using a range (eg. NNN-NNN). If a range was specified, consider making it a wider range, or a different range.

4) (Client only) If the client_port option was used, verify that there are no firewall restrictions on the use of the port.

Unable to clean up the socket layer.

Item	Value
Error code	61
Constant	SOCKET_CLEANUP (Java) STREAM_ERROR_- SOCKET_CLEANUP (C/C++) ulStreamErrorSocket- Cleanup (Visual Basic)

Probable cause The network layer was unable to clean up the socket layer. This error should only occur after all connections are finished, so no current connections should be affected.

Unable to close a socket.

Item	Value
Error code	62
Constant	SOCKET_CLOSE (Java) STREAM_ERROR_SOCKET_- CLOSE (C/C++) ulStreamErrorSocketClose (Visual Basic)

Probable cause The network layer was unable to close a socket. The network session may or may not have terminated prematurely, due to pending writes that were not flushed. Check the following:

- 1) The other side of the network connection had any errors.
- 2) The other side of the connection is running normally.
- 3) The machine is still connected to the network, and the network is responsive.

Unable to connect a socket.

Item	Value
Error code	63
Constant	SOCKET_CONNECT (Java) STREAM_ERROR_- SOCKET_CONNECT (C/C++) ulStreamErrorSocket- Connect (Visual Basic)

Probable cause The network layer was unable to connect a socket. Check the following:

- 1) The machine is connected to the network.
- 2) The socket layer is properly initialized.

-
- 3) The correct host machine and port were specified.
 - 4) The host server is running normally and listening on the correct port.
 - 5) The host machine is listening for the proper socket type (TCP/IP vs. UDP).
 - 6) If the client_port option was used, verify that there are no firewall restrictions on the use of the port.
 - 7) If the device has a limit on the number of open sockets, verify that the limit has not been reached.
 - 8) There are enough system resources available. Free up system resources if they are running low.

Unable to create a TCP/IP socket.

Item	Value
Error code	58
Constant	SOCKET_CREATE_TCPIP (Java) STREAM_ERROR_- SOCKET_CREATE_TCPIP (C/C++) ulStreamErrorSocketCreateTcpip (Visual Basic)

- Probable cause
- The network layer was unable to create a TCP/IP socket. Check the following:
- 1) The machine is connected to the network.
 - 2) The socket layer is properly initialized.
 - 5) If the device has a limit on the number of open sockets, verify that the limit has not been reached.
 - 6) There are enough system resources available. Free up system resources if they are running low.

Unable to create a UDP socket.

Item	Value
Error code	59
Constant	SOCKET_CREATE_UDP (Java) STREAM_ERROR_- SOCKET_CREATE_UDP (C/C++) ulStreamErrorSocketCreateUdp (Visual Basic)

- Probable cause
- The network layer was unable to create a UDP socket. Check the following:

- 1) The machine is connected to the network.
- 2) The socket layer is properly initialized.
- 3) If the `client_port` option was used, verify that the given port isn't already in use. If only one client port was specified, consider using a range (eg. NNN-NNN). If a range was specified, consider making it a wider range, or a different range.
- 4) If the `client_port` option was used, verify that there are no firewall restrictions on the use of the port.
- 5) If the device has a limit on the number of open sockets, verify that the limit has not been reached.
- 6) There are enough system resources available. Free up system resources if they are running low.

Unable to get host by address.

Item	Value
Error code	72
Constant	SOCKET_GET_HOST_BY_ADDR (Java) STREAM_ERROR_SOCKET_GET_HOST_BY_ADDR (C/C++) ulStreamErrorSocketGetHostByAddr (Visual Basic)

Probable cause The network layer was unable to get the name of a host using its IP address. At the time of publication, this error was defined but not used.

Unable to get a socket's local name.

Item	Value
Error code	64
Constant	SOCKET_GET_NAME (Java) STREAM_ERROR_SOCKET_GET_NAME (C/C++) ulStreamErrorSocketGetName (Visual Basic)

Probable cause The network layer was unable to determine a socket's local name. In a TCP/IP connection, each end of the connection has a socket exclusively attached to a port. A socket's local name includes this port number, which is assigned by the network at connection time. Check the following:

- 1) The machine is still connected to the network, and the network is responsive.

- 2) The other side of the connection is running normally.
- 3) There are enough system resources available. Free up system resources if they are running low.

Unable to get socket option number %1!s!.

Item	Value
Error code	65
Constant	SOCKET_GET_OPTION (Java) STREAM_ERROR_SOCKET_GET_OPTION (C/C++) ulStreamErrorSocketGetOption (Visual Basic)
Parameter 1	The socket option being retrieved.

Probable cause

The network layer was unable to get a socket option. This error may be the first indication that a connection has been lost. Check the following:

- 1) The machine is still connected to the network, and the network is responsive.
- 2) The other side of the connection is running normally.
- 3) There are enough system resources available. Free up system resources if they are running low.

The host name '%1!s!' could not be found.

Item	Value
Error code	57
Constant	SOCKET_HOST_NAME_NOT_FOUND (Java) STREAM_ERROR_SOCKET_HOST_NAME_NOT_FOUND (C/C++) ulStreamErrorSocketHostNameNotFound (Visual Basic)
Parameter 1	The name of the host.

Probable cause

The given host name could not be found. Check the following:

- 1) The host name was correctly specified.
- 2) The host is accessible. Many systems include a “ping” utility that can be used to verify access to a named host.
- 3) The Domain Name Server (DNS), or its equivalent, is available. If the DNS is not available, try specifying the host’s IP number (eg.

NNN.NNN.NNN.NNN) instead of the host name.

4) The HOSTS file contains an entry that maps the host name to an IP number.

Unable to listen on a socket. The backlog is %1!s!.

Item	Value
Error code	67
Constant	SOCKET_LISTEN (Java) STREAM_ERROR_- SOCKET_LISTEN (C/C++) ulStreamErrorSocketListen (Visual Basic)
Parameter 1	The requested listener backlog.

Probable cause

The server is unable to listen on a socket. The backlog refers to the maximum number of queued connection requests that may be pending at any given time. Check the following:

- 1) The machine is still connected to the network, and the network is responsive.
- 2) There are no firewall or other restrictions preventing a socket listener from running on the current machine.
- 3) The backlog setting is within the limit, if any, on the machine.
- 4) There are enough system resources available. Free up system resources if they are running low.

Invalid liveness timeout value %1!s!. The value must be between zero and 65535.

Item	Value
Error code	200
Constant	SOCKET_LIVENESS_OUT_OF_RANGE (Java) STREAM_ERROR_SOCKET_LIVENESS_OUT_OF_- RANGE (C/C++) ulStreamErrorSocketLivenessOut- OfRange (Visual Basic)
Parameter 1	The liveness timeout value.

Probable cause

An invalid liveness timeout value was specified. The liveness timeout value must be an integer between zero and 65535.

Unable to determine localhost.

Item	Value
Error code	71
Constant	SOCKET_LOCALHOST_NAME_NOT_FOUND (Java) STREAM_ERROR_SOCKET_LOCALHOST_NAME_NOT_FOUND (C/C++) ulStreamErrorSocketLocalhostNameNotFound (Visual Basic)

- Probable cause The network layer was unable to determine the IP address of “localhost”. Check the following:
- 1) The Domain Name Server (DNS), or its equivalent, is available. If the DNS is not available, try explicitly specifying the localhost IP number (usually 127.0.0.1) instead.
 - 2) The HOSTS file contains an entry that maps the “localhost” name to an IP number.
 - 3) There are enough system resources available. Free up system resources if they are running low.

Invalid port number %1s!. The value must be between zero and 65535.

Item	Value
Error code	74
Constant	SOCKET_PORT_OUT_OF_RANGE (Java) STREAM_ERROR_SOCKET_PORT_OUT_OF_RANGE (C/C++) ulStreamErrorSocketPortOutOfRange (Visual Basic)
Parameter 1	The port number.

- Probable cause An invalid port number was specified. The port number must be an integer between zero and 65535.

Unable to select a socket status.

Item	Value
Error code	69
Constant	SOCKET_SELECT (Java) STREAM_ERROR_SOCKET_SELECT (C/C++) ulStreamErrorSocketSelect (Visual Basic)

Probable cause The network layer encountered an error attempting to wait for a socket to be ready for reading or writing. Check the following:

- 1) The machine is connected to the network, and the network is responsive.
- 2) The other side of the connection is running normally.
- 3) There are enough system resources available. Free up system resources if they are running low.

Unable to set socket option number %1!s!.

Item	Value
Error code	66
Constant	SOCKET_SET_OPTION (Java) STREAM_ERROR_- SOCKET_SET_OPTION (C/C++) ulStreamErrorSocket- SetOption (Visual Basic)
Parameter 1	The socket option being set.

Probable cause The network layer was unable to set a socket option. This error may be the first indication that a connection has been lost. Check the following:

- 1) The machine is still connected to the network, and the network is responsive.
- 2) The other side of the connection is running normally.
- 3) There are enough system resources available. Free up system resources if they are running low.

Unable to shut down a socket.

Item	Value
Error code	68
Constant	SOCKET_SHUTDOWN (Java) STREAM_ERROR_- SOCKET_SHUTDOWN (C/C++) ulStreamErrorSock- etShutdown (Visual Basic)

Probable cause The network layer was unable to shut down a socket. Check the following:

- 1) The machine is connected to the network, and the network is responsive.
- 2) The other side of the connection is running normally.
- 3) There are enough system resources available. Free up system resources if they are running low.

Unable to initialize the sockets layer.

Item	Value
Error code	70
Constant	SOCKET_STARTUP (Java) STREAM_ERROR_- SOCKET_STARTUP (C/C++) ulStreamErrorSocket- Startup (Visual Basic)

Probable cause The network layer was unable to initialize the socket layer. Check the following:

- 1) The sockets layer is properly installed. The correct network interface library must be present and accessible.
- 2) The machine is connected to the network, and the network is responsive.
- 3) There are enough system resources available. Free up system resources if they are running low.

The operation would cause blocking.

Item	Value
Error code	13
Constant	WOULD_BLOCK (Java) STREAM_ERROR_WOULD_- BLOCK (C/C++) ulStreamErrorWouldBlock (Visual Ba- sic)

Probable cause A requested operation would block where blocking is undesirable or unexpected.

Unable to write %1!s! bytes.

Item	Value
Error code	9
Constant	WRITE (Java) STREAM_ERROR_WRITE (C/C++) ul- StreamErrorWrite (Visual Basic)
Parameter 1	The number of bytes that could not be written.

Probable cause Unable to write the given number of bytes to the network layer. Note that writes may occur as part of any larger network operation. For example, some network layers have sub-layers that perform several reads and writes as part

of a basic operation in the upper layer.

The cause of a write error is usually one of the following:

1) The network had a problem that caused the write to fail.

Reconnect and retry the operation.

2) The connection timed out.

Reconnect and retry the operation.

3) The other side of the connection cleanly terminated the connection.

Consult the client and/or server logs for errors that indicate why the connection has been dropped.

Consult the output-log errors and fix the cause, then retry the operation.

4) The process at the other side of the connection was aborted.

Consult the client and/or server output logs for errors that indicate why the process was aborted.

If the process was shut down by other than normal means, there may not be any errors in its output log.

Reconnect and retry the operation.

5) The system is low on resources, and cannot perform the write.

Free up system resources, reconnect and retry the operation. If subsequent retry attempts fail, consult your network administrator.

Timed out trying to write %1s! bytes.

Item	Value
Error code	202
Constant	WRITE_TIMEOUT (Java) STREAM_ERROR_WRITE_TIMEOUT (C/C++) ulStreamErrorWriteTimeout (Visual Basic)
Parameter 1	The number of bytes that could not be written.

Probable cause

Unable to write the given number of bytes to the network layer in the given time.

Check that the network is functioning correctly, and that the receiving application is still running.

CHAPTER 13

MobiLink Synchronization Server Error Messages

About this chapter

This chapter lists MobiLink synchronization server communication errors, as well as their probable causes.

The error messages are written to the MobiLink synchronization server message log.

Contents

Topic:	page
MobiLink synchronization server error messages sorted by code	400
MobiLink synchronization server error messages sorted message	406
MobiLink synchronization server error descriptions	412

MobiLink synchronization server error messages sorted by code

Error code	Error message
–10094	“Expecting %1!ld! authentication parameter(s) from client, but received %2!ld! for script %3!s!” on page 416
–10093	“There is no download data script defined for table: %1!s!. If you want to be able to synchronize anyway, with the risk of potentially losing download data, use the -fr switch” on page 427
–10092	“There is no upload data script defined for table: %1!s!. If you want to be able to synchronize anyway, with the risk of potentially losing upload data, use the -fr switch” on page 427
–10091	“This connection will be abandoned due to previous errors” on page 428
–10090	“The client cannot find the consolidated progress offset from the client transaction log(s)” on page 424
–10089	“Client is unable to process truncate table request for table ‘%1!s!’” on page 415
–10088	“Unable to load entry points from dll: ‘%1!s!’” on page 434
–10087	“Version mismatch with dll : ‘%1!s!’\nExpected version: %2!d! got version: %3!d!” on page 436
–10086	“Cannot load dll: ‘%1!s!’ for Script Language: ‘%2!s!’” on page 414
–10085	“LANG: %1!s! = Failed to allocate database connection” on page 419
–10084	“LANG: %1!s! - Failed to attach worker thread” on page 419
–10083	“Unable to delete user name ‘%1!s!’ from the ml_user_ table” on page 430
–10082	“Unable to initialize the resource DLL ‘%1!s!’” on page 432

Error code	Error message
-10081	"The MobiLink synchronization server DLL version does not match the data layer DLL version" on page 423
-10080	"Unable to execute script '%1s!'" on page 431
-10079	"The length of the name of a publication, table, or column cannot be retrieved from the upload stream" on page 424
-10078	"The publication, table, or column name received from the client is too long: the length is %1d!" on page 425
-10077	"The MobiLink synchronization server was unable to modify the error message using the modify_error_message script" on page 424
-10076	"The MobiLink synchronization server was unable to calculate the timestamp precision on the consolidated database using the ml_scripts_modified table. Timestamp precision related warnings will not be generated" on page 423
-10075	"Required ODBC function %1s! is not supported by the driver" on page 422
-10074	"Unable to update table '%1s!' using %2s!" on page 435
-10073	"Unable to delete from table '%1s!' using %2s!" on page 430
-10072	"Unable to insert into table '%1s!' using %2s!" on page 433
-10071	"Unable to fetch from table '%1s!' using %2s!" on page 431
-10070	"No server connection string specified" on page 420
-10069	"Unable to initialize consolidated database interface" on page 432
-10068	"Unable to initialize authentication subsystem" on page 432
-10067	"Unable to allocate a connection" on page 429
-10066	"Unable to initialize ODBC" on page 432
-10065	"Unable to COMMIT Transaction: %1s! – Attempting to ROLLBACK" on page 428

Error code	Error message
-10064	“Unable to ROLLBACK Transaction: %1!s!” on page 428
-10063	“An error occurred while uploading an updated row into table ‘%1!s!’. The updated column values are as follows:” on page 413
-10062	“An error occurred while uploading a deleted row into table ‘%1!s!’. The deleted column values are as follows:” on page 413
-10061	“An error occurred while uploading an insert row into table ‘%1!s!’. The inserted column values are as follows:” on page 413
-10060	“Memory allocation failed” on page 420
-10059	“A protocol error occurred when attempting to retrieve the remote client’s synchronization log” on page 412
-10058	“Unable to open %1!s!” on page 434
-10057	“Invalid password for user %1!s!” on page 419
-10056	“User name ‘%1!s!’ not found in the ml_user table” on page 435
-10055	“Unable to authenticate user %1!s!” on page 429
-10054	“Unable to insert user name ‘%1!s!’ into the ml_user table” on page 433
-10053	“The user name ‘%1!s!’ is already synchronizing. Concurrent synchronizations using the same user name are not allowed” on page 426
-10052	“The %1!s! script returned %2!ld!” on page 423
-10051	“Internal error: wrong function ‘%1!s!’ called. Please contact technical support” on page 419
-10050	“Expecting %1!ld! columns in cursor, but found %2!ld!” on page 417
-10049	“Too many bind parameters in script (expecting %1!ld! but found %2!ld!): %3!s!” on page 428
-10048	“Expecting at least %1!ld! parameters in script, but only found %2!ld!: %3!s!” on page 417

Error code	Error message
-10047	“Expecting %1!d! parameters in script, but only found %2!d!: %3!s!” on page 417
-10046	“Unable to allocate an input/output cursor” on page 429
-10045	“Cannot directly determine the name of the table referenced by the cursor. The table name is required for inserts, updates, and deletes when using the Microsoft ODBC Cursor Library” on page 414
-10044	“INTERNAL ERROR: occurred while storing a BLOB – write” on page 418
-10043	“INTERNAL ERROR: occurred while retrieving a BLOB – zero length” on page 418
-10042	“INTERNAL ERROR: occurred while retrieving a BLOB – null” on page 418
-10041	“INTERNAL ERROR: occurred while retrieving a BLOB – read” on page 418
-10040	“Extraneous data found in upload stream” on page 417
-10039	“Scripts cannot be defined as NULL” on page 422
-10038	“A downloaded value for table %1!s! (column #%2!d!) was either too big or invalid for the remote schema type” on page 412
-10037	“Unable to launch the command: (%1!s!). The system error code is %2!d!” on page 433
-10036	“Download stream encountered error in remote database” on page 416
-10035	“Download failed with client error %1!d!” on page 415
-10034	“No download confirmation from remote database” on page 420
-10033	“The row is too big. The size (%1!d! bytes) exceeds the maximum allowable size (%2!d! bytes)” on page 425
-10032	“Upload failed with client error %1!d!” on page 435
-10031	“An error occurred when trying to store progress information in the consolidated database” on page 413

Error code	Error message
-10030	“A network read failed. Unable to read data from the remote client” on page 412
-10029	“Attempt to set non-null column to null” on page 414
-10028	“Unable to connect to the consolidated database. Aborting the synchronization” on page 429
-10027	“Unable to generate scripts for version ‘%1!s!’” on page 431
-10026	“The upload stream is too short: should be at least %1!d! bytes, but received %2!d! bytes” on page 426
-10025	“The %1!s! cursor is unexpectedly undefined” on page 423
-10024	“Unrecognized domain id %1!d!” on page 435
-10023	“The remote database may have been restored from backup, or perhaps user name ‘%1!s!’ is being used by different remote databases. Set ml_user.commit_state to zero to re-enable synchronizations for this user” on page 425
-10022	“The synchronization sequence number stored in ml_user.commit_state is negative. Set this value to zero (0) to re-enable synchronizations for user ‘%1!s!’” on page 426
-10021	“Unable to retry the current transaction after deadlock in the consolidated database. The retry limit has been reached” on page 434
-10020	“Unable to flush scripts” on page 431
-10019	“Error fetching table script %1!s!.%2!s!” on page 416
-10018	“Error fetching connection script %1!s!” on page 416
-10017	“Protocol error: there is no publication that contains table ‘%1!s!’” on page 422
-10016	“Cannot convert ‘%1!s!’ to Unicode” on page 414
-10015	“Protocol error: client requests an unsupported capability (%1!s!)” on page 421
-10014	“Protocol error: an invalid timestamp precision of %1!d! was sent from the remote” on page 421

Error code	Error message
-10013	“Version ‘%1!s!’ not found in the ml_script_version table. Cannot synchronize” on page 436
-10012	“There are no registered script versions. Unable to synchronize a client created prior to version 7.0.0” on page 427
-10011	“Unable to determine the remote version” on page 430
-10010	“Unable to determine the remote user password” on page 430
-10009	“Unable to determine the remote user name” on page 430
-10008	“Unable to load UNILIB collation expansion factor: error %1!d!” on page 434
-10007	“Unable to load UNILIB collation %1!d!: error %2!d!” on page 433
-10006	“Collation not supported by this server” on page 415
-10005	“Old versions of MobiLink clients cannot ping the MobiLink synchronization server” on page 421
-10004	“Protocol version mismatch” on page 422
-10003	“Memory allocation failed, attempted to allocate %1!lu! bytes” on page 420
-10002	“Consolidated database server or ODBC error: %1!s!” on page 415
-10001	“Protocol error” on page 421
0	“No error or unknown error” on page 420

MobiLink synchronization server error messages sorted message

Error code	Error message
-10038	"A downloaded value for table %1!s! (column #%2!ld!) was either too big or invalid for the remote schema type" on page 412
-10030	"A network read failed. Unable to read data from the remote client" on page 412
-10059	"A protocol error occurred when attempting to retrieve the remote client's synchronization log" on page 412
-10031	"An error occurred when trying to store progress information in the consolidated database" on page 413
-10062	"An error occurred while uploading a deleted row into table '%1!s!'. The deleted column values are as follows:" on page 413
-10061	"An error occurred while uploading an insert row into table '%1!s!'. The inserted column values are as follows:" on page 413
-10063	"An error occurred while uploading an updated row into table '%1!s!'. The updated column values are as follows:" on page 413
-10029	"Attempt to set non-null column to null" on page 414
-10016	"Cannot convert '%1!s!' to Unicode" on page 414
-10045	"Cannot directly determine the name of the table referenced by the cursor. The table name is required for inserts, updates, and deletes when using the Microsoft ODBC Cursor Library" on page 414
-10086	"Cannot load dll: '%1!s!' for Script Language: '%2!s!'" on page 414
-10089	"Client is unable to process truncate table request for table '%1!s!'" on page 415
-10006	"Collation not supported by this server" on page 415
-10002	"Consolidated database server or ODBC error: %1!s!" on page 415

Error code	Error message
-10035	“Download failed with client error %1!d!” on page 415
-10036	“Download stream encountered error in remote database” on page 416
-10018	“Error fetching connection script %1!s!” on page 416
-10019	“Error fetching table script %1!s!.%2!s!” on page 416
-10094	“Expecting %1!ld! authentication parameter(s) from client, but received %2!ld! for script %3!s!” on page 416
-10050	“Expecting %1!ld! columns in cursor, but found %2!ld!” on page 417
-10047	“Expecting %1!ld! parameters in script, but only found %2!ld!: %3!s!” on page 417
-10048	“Expecting at least %1!ld! parameters in script, but only found %2!ld!: %3!s!” on page 417
-10040	“Extraneous data found in upload stream” on page 417
-10042	“INTERNAL ERROR: occurred while retrieving a BLOB – null” on page 418
-10041	“INTERNAL ERROR: occurred while retrieving a BLOB – read” on page 418
-10043	“INTERNAL ERROR: occurred while retrieving a BLOB – zero length” on page 418
-10044	“INTERNAL ERROR: occurred while storing a BLOB – write” on page 418
-10051	“Internal error: wrong function ‘%1!s!’ called. Please contact technical support” on page 419
-10057	“Invalid password for user %1!s!” on page 419
-10084	“LANG: %1!s! - Failed to attach worker thread” on page 419
-10085	“LANG: %1!s! = Failed to allocate database connection” on page 419
-10060	“Memory allocation failed” on page 420
-10003	“Memory allocation failed, attempted to allocate %1!lu! bytes” on page 420

Error code	Error message
-10034	"No download confirmation from remote database" on page 420
0	"No error or unknown error" on page 420
-10070	"No server connection string specified" on page 420
-10005	"Old versions of MobiLink clients cannot ping the MobiLink synchronization server" on page 421
-10001	"Protocol error" on page 421
-10014	"Protocol error: an invalid timestamp precision of %!d! was sent from the remote" on page 421
-10015	"Protocol error: client requests an unsupported capability (%!s!)" on page 421
-10017	"Protocol error: there is no publication that contains table '%!s!'" on page 422
-10004	"Protocol version mismatch" on page 422
-10075	"Required ODBC function %!s! is not supported by the driver" on page 422
-10039	"Scripts cannot be defined as NULL" on page 422
-10025	"The %!s! cursor is unexpectedly undefined" on page 423
-10052	"The %!s! script returned %!d!" on page 423
-10081	"The MobiLink synchronization server DLL version does not match the data layer DLL version" on page 423
-10076	"The MobiLink synchronization server was unable to calculate the timestamp precision on the consolidated database using the ml_scripts_modified table. Timestamp precision related warnings will not be generated" on page 423
-10077	"The MobiLink synchronization server was unable to modify the error message using the modify_error_message script" on page 424
-10090	"The client cannot find the consolidated progress offset from the client transaction log(s)" on page 424
-10079	"The length of the name of a publication, table, or column cannot be retrieved from the upload stream" on page 424

Error code	Error message
-10078	“The publication, table, or column name received from the client is too long: the length is %1!d!” on page 425
-10023	“The remote database may have been restored from backup, or perhaps user name ‘%1!s!’ is being used by different remote databases. Set ml_user.commit_state to zero to re-enable synchronizations for this user” on page 425
-10033	“The row is too big. The size (%1!d! bytes) exceeds the maximum allowable size (%2!d! bytes)” on page 425
-10022	“The synchronization sequence number stored in ml_user.commit_state is negative. Set this value to zero (0) to re-enable synchronizations for user ‘%1!s!’” on page 426
-10026	“The upload stream is too short: should be at least %1!d! bytes, but received %2!d! bytes” on page 426
-10053	“The user name ‘%1!s!’ is already synchronizing. Concurrent synchronizations using the same user name are not allowed” on page 426
-10012	“There are no registered script versions. Unable to synchronize a client created prior to version 7.0.0” on page 427
-10093	“There is no download data script defined for table: %1!s!. If you want to be able to synchronize anyway, with the risk of potentially losing download data, use the -fr switch” on page 427
-10092	“There is no upload data script defined for table: %1!s!. If you want to be able to synchronize anyway, with the risk of potentially losing upload data, use the -fr switch” on page 427
-10091	“This connection will be abandoned due to previous errors” on page 428
-10049	“Too many bind parameters in script (expecting %1!d! but found %2!d!); %3!s!” on page 428
-10065	“Unable to COMMIT Transaction: %1!s! – Attempting to ROLLBACK” on page 428
-10064	“Unable to ROLLBACK Transaction: %1!s!” on page 428

Error code	Error message
-10067	“Unable to allocate a connection” on page 429
-10046	“Unable to allocate an input/output cursor” on page 429
-10055	“Unable to authenticate user ‘%1!s!’” on page 429
-10028	“Unable to connect to the consolidated database. Aborting the synchronization” on page 429
-10073	“Unable to delete from table ‘%1!s!’ using ‘%2!s!’” on page 430
-10083	“Unable to delete user name ‘%1!s!’ from the ml_user_table” on page 430
-10009	“Unable to determine the remote user name” on page 430
-10010	“Unable to determine the remote user password” on page 430
-10011	“Unable to determine the remote version” on page 430
-10080	“Unable to execute script ‘%1!s!’” on page 431
-10071	“Unable to fetch from table ‘%1!s!’ using ‘%2!s!’” on page 431
-10020	“Unable to flush scripts” on page 431
-10027	“Unable to generate scripts for version ‘%1!s!’” on page 431
-10066	“Unable to initialize ODBC” on page 432
-10068	“Unable to initialize authentication subsystem” on page 432
-10069	“Unable to initialize consolidated database interface” on page 432
-10082	“Unable to initialize the resource DLL ‘%1!s!’” on page 432
-10072	“Unable to insert into table ‘%1!s!’ using ‘%2!s!’” on page 433
-10054	“Unable to insert user name ‘%1!s!’ into the ml_user table” on page 433
-10037	“Unable to launch the command: (%1!s!). The system error code is %2!d!” on page 433

Error code	Error message
-10007	“Unable to load UNILIB collation %1!d!: error %2!d!” on page 433
-10008	“Unable to load UNILIB collation expansion factor: error %1!d!” on page 434
-10088	“Unable to load entry points from dll: ‘%1!s!’” on page 434
-10058	“Unable to open %1!s!” on page 434
-10021	“Unable to retry the current transaction after deadlock in the consolidated database. The retry limit has been reached” on page 434
-10074	“Unable to update table ‘%1!s!’ using %2!s!” on page 435
-10024	“Unrecognized domain id %1!d!” on page 435
-10032	“Upload failed with client error %1!d!” on page 435
-10056	“User name ‘%1!s!’ not found in the ml_user table” on page 435
-10013	“Version ‘%1!s!’ not found in the ml_script_version table. Cannot synchronize” on page 436
-10087	“Version mismatch with dll : ‘%1!s!’\nExpected version: %2!d! got version: %3!d!” on page 436

MobiLink synchronization server error descriptions

This section provides a full listing of error messages and descriptions.

Errors with an ODBC state marked “handled by ODBC driver” are not returned to ODBC applications, as the ODBC driver carries out the required actions.

A downloaded value for table %1!s! (column #%2!ld!) was either too big or invalid for the remote schema type

Item	Value
Error code	–10038
Parameter 1	Table name and column index.

Probable cause The column width for the given table may not be defined consistently in the consolidated and remote databases. Please check the table definition.

A network read failed. Unable to read data from the remote client

Item	Value
Error code	–10030

Probable cause The MobiLink synchronization server was unable to complete a network read. Please check the network.

A protocol error occurred when attempting to retrieve the remote client’s synchronization log

Item	Value
Error code	–10059

Probable cause There was a protocol error when the MobiLink synchronization server was retrieving the client error file. Please make sure that the client is supported by your version of the MobiLink synchronization server.

An error occurred when trying to store progress information in the consolidated database

Item	Value
Error code	–10031

Probable cause The MobiLink synchronization server is unable to save the synchronization status into the consolidated database. Please make sure that the database server is running and the network is okay.

An error occurred while uploading a deleted row into table '%1!s!'. The deleted column values are as follows:

Item	Value
Error code	–10062
Parameter 1	The script name.

Probable cause A failure occurred when the MobiLink synchronization server was uploading a deleted row into the given table in the consolidated database.

An error occurred while uploading an insert row into table '%1!s!'. The inserted column values are as follows:

Item	Value
Error code	–10061
Parameter 1	The script name.

Probable cause A failure occurred when the MobiLink synchronization server was uploading an inserted row into the given table in the consolidated database.

An error occurred while uploading an updated row into table '%1!s!'. The updated column values are as follows:

Item	Value
Error code	–10063
Parameter 1	The script name.

Probable cause A failure occurred when the MobiLink synchronization server was uploading an updated row into the given table in the consolidated database.

Attempt to set non-null column to null

Item	Value
Error code	-10029

Probable cause The MobiLink synchronization server attempted to download a null into a non-nullable column.

Cannot convert '%1!s!' to Unicode

Item	Value
Error code	-10016
Parameter 1	String to be converted.

Probable cause The MobiLink synchronization server was not able to convert the given string to Unicode using Unilib.

Cannot directly determine the name of the table referenced by the cursor. The table name is required for inserts, updates, and deletes when using the Microsoft ODBC Cursor Library

Item	Value
Error code	-10045

Probable cause The MobiLink synchronization server cannot directly determine the name of the table referenced by the upload cursor. The table name is required for inserts, updates, and deletes when using the Microsoft ODBC Cursor Library.

Cannot load dll: '%1!s!' for Script Language: '%2!s!'

Item	Value
Error code	-10086
Parameter 1	The DLL name and the script language name.

Probable cause Please make sure that the script language is valid. Currently the script languages supported by the MobiLink synchronization server are SQL (sql), Java (java) and .NET (dnet).

Client is unable to process truncate table request for table '%1!s!'

Item	Value
Error code	-10089
Parameter 1	The name of a table.

Probable cause The download_delete_cursor script is requesting that the table be truncated. The client needs to be updated to a newer version in order to process this action.

Collation not supported by this server

Item	Value
Error code	-10006

Probable cause Old clients were trying to send Unicode-based strings (strings in UTF8). Please upgrade the client.

Consolidated database server or ODBC error: %1!s!

Item	Value
Error code	-10002
Parameter 1	The actual error message sent by the database server or ODBC driver.

Probable cause This may be a SQL error such as a syntax error.

Download failed with client error %1!d!

Item	Value
Error code	-10035
Parameter 1	An error number sent by the client.

Probable cause The MobiLink synchronization server aborts the synchronization when the client indicates there is a problem on the remote site during download.

Download stream encountered error in remote database

Item	Value
Error code	–10036

Probable cause The client indicates that the download failed.

Error fetching connection script %1!s!

Item	Value
Error code	–10018
Parameter 1	Connection script name.

Probable cause The MobiLink synchronization server was not able to refresh connection scripts. Please make sure that the database server is running and the network is okay.

Error fetching table script %1!s!.%2!s!

Item	Value
Error code	–10019
Parameter 1	Table and script name.

Probable cause The MobiLink synchronization server was not able to refresh table scripts. Please make sure that the database server is running and the network is okay.

Expecting %1!ld! authentication parameter(s) from client, but received %2!ld! for script %3!s!

Item	Value
Error code	–10094
Parameter 1	The number of parameters expected, the number of parameters passed up from the client, and the script that needs the parameters.

Probable cause The number of authentication parameters received from the client does not match the number expected. The number of client parameters should be two less than the number required by the authenticate_parameters script.

Expecting %1!ld! columns in cursor, but found %2!ld!

Item	Value
Error code	-10050
Parameter 1	The number of columns expected and the number of columns found.

Probable cause The number of parameters found in the upload or download script does not match the number of columns or the number of primary key columns for the given table. Please check the number of parameters for the given script.

Expecting %1!ld! parameters in script, but only found %2!ld!: %3!s!

Item	Value
Error code	-10047
Parameter 1	The number of parameters expected, the number of parameters found, and the script name.

Probable cause There are too many parameters found in the given script. Please check the number of parameters for the given script.

Expecting at least %1!ld! parameters in script, but only found %2!ld!: %3!s!

Item	Value
Error code	-10048
Parameter 1	The minimum number of parameters expected, the number of parameters found, and the script name.

Probable cause There are not enough parameters found in the given script. Please check the number of parameters for the given script.

Extraneous data found in upload stream

Item	Value
Error code	-10040

Probable cause The client has sent extraneous data to the MobiLink synchronization server. Please check that the client is supported by your MobiLink synchronization

server.

INTERNAL ERROR: occurred while retrieving a BLOB – null

Item	Value
Error code	–10042

Probable cause The MobiLink synchronization server cannot retrieve the upload data from memory or a temporary file. This is an internal error. Please contact technical support.

INTERNAL ERROR: occurred while retrieving a BLOB – read

Item	Value
Error code	–10041

Probable cause The MobiLink synchronization server cannot retrieve the upload data from memory or a temporary file. This is an internal error. Please contact technical support.

INTERNAL ERROR: occurred while retrieving a BLOB – zero length

Item	Value
Error code	–10043

Probable cause The MobiLink synchronization server cannot retrieve the upload data from memory or a temporary file. This is an internal error. Please contact technical support.

INTERNAL ERROR: occurred while storing a BLOB – write

Item	Value
Error code	–10044

Probable cause The MobiLink synchronization server cannot store the upload data to memory or a temporary file. This is an internal error. Please contact technical support.

Internal error: wrong function ‘%1!s!’ called. Please contact technical support

Item	Value
Error code	–10051
Parameter 1	The function name.

Probable cause This is an internal error. Please contact technical support.

Invalid password for user %1!s!

Item	Value
Error code	–10057
Parameter 1	The user name.

Probable cause The password sent up from the remote is invalid for the given user. Please note: passwords are case sensitive.

LANG: %1!s! - Failed to attach worker thread

Item	Value
Error code	–10084
Parameter 1	The name of the script language.

Probable cause A MobiLink synchronization server worker thread could not attach itself to the DLL or shared object used to process the given script language. Please make sure that the MobiLink synchronization server installation contains all of the required DLLs or shared objects.

LANG: %1!s! = Failed to allocate database connection

Item	Value
Error code	–10085
Parameter 1	The name of the script language.

Probable cause A connection cannot be made to the database server.

Memory allocation failed

Item	Value
Error code	-10060

Probable cause Your system is running out of memory. You may need to close some applications or add more memory to your system.

Memory allocation failed, attempted to allocate %1!lu! bytes

Item	Value
Error code	-10003
Parameter 1	The number of bytes it was trying to allocate.

Probable cause Your system is running out of memory. You may need to close some applications or add more memory to your system.

No download confirmation from remote database

Item	Value
Error code	-10034

Probable cause The client attempted to send a confirmation status to the MobiLink synchronization server after download. The MobiLink synchronization server did not receive this confirmation. This could happen if the synchronization is interrupted or if there is a network problem.

No error or unknown error

Item	Value
Error code	0

Probable cause This code indicates there was either no error or an unknown error. The MobiLink synchronization server did not assign an error number.

No server connection string specified

Item	Value
Error code	-10070

Probable cause There is no server connection string specified in the MobiLink synchronization server command line. Please specify the connection string in the MobiLink synchronization server command line using the -c option.

Old versions of MobiLink clients cannot ping the MobiLink synchronization server

Item	Value
Error code	-10005

Probable cause Ping is available only from version 8.0.1 and up.

Protocol error

Item	Value
Error code	-10001

Probable cause The MobiLink synchronization server does not understand the request sent by the client. This could happen if the client is newer than the MobiLink synchronization server.

Protocol error: an invalid timestamp precision of %1!d! was sent from the remote

Item	Value
Error code	-10014
Parameter 1	Timestamp precision sent by the client.

Probable cause The timestamp precision must be greater than 0 and less than 6.

Protocol error: client requests an unsupported capability (%1!s!)

Item	Value
Error code	-10015
Parameter 1	Capability bit.

Probable cause The MobiLink synchronization server does not support the capability requested by the client. Please make sure that you are not using a newer client to talk to an older MobiLink synchronization server.

Protocol error: there is no publication that contains table '%1!s!'

Item	Value
Error code	-10017
Parameter 1	Table name.

Probable cause A table is requested for synchronization by the client. However the table is not in any publication. Please make sure that the table was not dropped in the remote database.

Protocol version mismatch

Item	Value
Error code	-10004

Probable cause The MobiLink synchronization server is communicating with the client using different protocol versions. If the client sends a version that is not supported by the MobiLink synchronization server, it will give this error. Please make sure that you are not using a newer client (DBMLSync or an UltraLite application) to talk to an older version of the MobiLink synchronization server.

Required ODBC function %1!s! is not supported by the driver

Item	Value
Error code	-10075
Parameter 1	The ODBC function name.

Probable cause The MobiLink synchronization server was not able to find the function from the ODBC driver. Please start the MobiLink synchronization server with the recommended ODBC drivers.

Scripts cannot be defined as NULL

Item	Value
Error code	-10039

Probable cause Scripts cannot be defined as NULL or empty string. Please check the script definitions to make sure that no script is defined as NULL or empty string.

The %1!s! cursor is unexpectedly undefined

Item	Value
Error code	-10025
Parameter 1	Cursor script name.

Probable cause This is an internal error. Please contact technical support.

The %1!s! script returned %2!ld!

Item	Value
Error code	-10052
Parameter 1	The user authentication script name and the script returning value.

Probable cause The user authentication script returned a value greater than 3000.

The MobiLink synchronization server DLL version does not match the data layer DLL version

Item	Value
Error code	-10081

Probable cause The MobiLink synchronization server DLL that processes the upload and download data is not consistent with the version of the data layer DLL (ODBC) that is used to interact with the databases. Please check your MobiLink synchronization server installation.

The MobiLink synchronization server was unable to calculate the timestamp precision on the consolidated database using the ml_scripts_modified table. Timestamp precision related warnings will not be generated

Item	Value
Error code	-10076

The MobiLink synchronization server was unable to modify the error message using the modify_error_message script

Item	Value
Error code	-10077

Probable cause An error occurred during synchronization, but the MobiLink synchronization server was unable to modify the error message using the given script. Please check your modify_error_message script.

The client cannot find the consolidated progress offset from the client transaction log(s)

Item	Value
Error code	-10090

Probable cause When the progress offsets in the consolidated and the remote databases are different, the MobiLink synchronization server may ask the client to redo the synchronization again with the progress offset from the consolidated database. However the requested consolidated progress offset cannot be found in the transaction log(s) of the remote database because the old transaction logs have been deleted or the requested offset has not been created yet.

Please find the source that caused this problem, and then rerun dbmlsync using the -ra or -rb option.

The length of the name of a publication, table, or column cannot be retrieved from the upload stream

Item	Value
Error code	-10079

Probable cause The client sent the MobiLink synchronization server the lengths of the names of publications, tables, and table columns. Please check the network.

The publication, table, or column name received from the client is too long: the length is %1!d!

Item	Value
Error code	–10078
Parameter 1	The length of the name.

Probable cause The client sent the MobiLink synchronization server the lengths of the names of publications, tables, and table columns. These names should be less than 128 bytes long. Please check the names of publications, table, and columns.

The remote database may have been restored from backup, or perhaps user name ‘%1!s!’ is being used by different remote databases. Set ml_user.commit_state to zero to re-enable synchronizations for this user

Item	Value
Error code	–10023
Parameter 1	User name.

Probable cause Before doing synchronization, the MobiLink synchronization server compares the sequence number sent by the client with that stored in the consolidated database to see if they match. If a remote database is restored from a backup or the last synchronization was interrupted, the sequence number may be less than that in the consolidated. Please set ml_user.commit_state to zero to re-enable synchronizations for this user.

The row is too big. The size (%1!d! bytes) exceeds the maximum allowable size (%2!d! bytes)

Item	Value
Error code	–10033
Parameter 1	The actual row size to be downloaded and the maximum row size the
Parameter 2	client is willing to accept.

Probable cause An UltraLite application may have a row size limit. If the row size in the synchronization table in the consolidated database has exceeded this limit, the table cannot be downloaded. Please redesign your synchronization table.

The synchronization sequence number stored in ml_user.commit_state is negative. Set this value to zero (0) to re-enable synchronizations for user '%1!s!'

Item	Value
Error code	-10022
Parameter 1	User name.

Probable cause The sequence number stored in the ml_user table in the consolidated database is negative. This number is maintained by the MobiLink synchronization server. In most cases, please do not directly modify this number.

The upload stream is too short: should be at least %1!d! bytes, but received %2!d! bytes

Item	Value
Error code	-10026
Parameter 1	The number of bytes that are expected and the number of bytes that are actually received.

Probable cause The MobiLink synchronization server received a connection that did not send enough data. Please check the remote for communication problems. Also check for an agent that is mistakenly connecting to MobiLink synchronization server. Also check for an unfriendly agent attempting to disrupt the MobiLink synchronization server.

The user name '%1!s!' is already synchronizing. Concurrent synchronizations using the same user name are not allowed

Item	Value
Error code	-10053
Parameter 1	The user name.

Probable cause The given user name is already synchronizing. Please try the synchronization later.

There are no registered script versions. Unable to synchronize a client created prior to version 7.0.0

Item	Value
Error code	-10012

Probable cause The client-requested script version was not defined in the consolidated database. Please create the scripts for the specified version in the consolidated database or use the `-za` option in the MobiLink synchronization server command line to allow the MobiLink synchronization server to generate active scripts.

There is no download data script defined for table: %1!s!. If you want to be able to synchronize anyway, with the risk of potentially losing download data, use the `-fr` switch

Item	Value
Error code	-10093
Parameter 1	The name of a table.

Probable cause If there is no download data script for a table and synchronization is download only, there is a risk of potentially losing the download data. To prevent this situation, the synchronization is aborted. If you do not care about losing download data, you can use `-fr` to force the synchronization to continue.

There is no upload data script defined for table: %1!s!. If you want to be able to synchronize anyway, with the risk of potentially losing upload data, use the `-fr` switch

Item	Value
Error code	-10092
Parameter 1	The name of a table.

Probable cause If there is no upload data script for a table, there is a risk of potentially losing the upload data. To prevent this situation, the synchronization is aborted. If you do not care about losing upload data, you can use `-fr` to force the synchronization to continue.

This connection will be abandoned due to previous errors

Item	Value
Error code	-10091

Probable cause Due to the severity of error(s) encountered processing the upload stream, further work will be futile. This is probably due to an I/O error or a protocol problem.

Too many bind parameters in script (expecting %1!ld! but found %2!ld!): %3!s!

Item	Value
Error code	-10049
Parameter 1	The number of parameters expected, the number of parameters found, and the script name.

Probable cause There are too many parameters found in the given script. Please check the number of parameters for the given script.

Unable to COMMIT Transaction: %1!s! – Attempting to ROLLBACK

Item	Value
Error code	-10065
Parameter 1	The script name.

Probable cause The MobiLink synchronization server was not able to commit the transaction for the given script. Writing scripts to avoid deadlocks is always a good practice.

Unable to ROLLBACK Transaction: %1!s!

Item	Value
Error code	-10064
Parameter 1	The script name.

Probable cause The MobiLink synchronization server was not able to roll back the transaction.

Unable to allocate a connection

Item	Value
Error code	–10067

Probable cause The MobiLink synchronization server was not able to allocate a connection. Please make sure that there is enough memory to start the MobiLink synchronization server. Also, make sure that the database server is running, and that the user ID and password are valid.

Unable to allocate an input/output cursor

Item	Value
Error code	–10046

Probable cause A failure occurred when the MobiLink synchronization server was trying to allocate memory for upload table scripts. Your system is running out of memory. You may need to close some applications or add more memory to your system.

Unable to authenticate user %1!s!

Item	Value
Error code	–10055
Parameter 1	The user name.

Probable cause The MobiLink synchronization server was not able to authenticate the given user.

Unable to connect to the consolidated database. Aborting the synchronization

Item	Value
Error code	–10028

Probable cause The MobiLink synchronization server was not able to make a connection to the consolidated database server. Please make sure that the database server is running and the network is okay.

Unable to delete from table '%1!s!' using %2!s!

Item	Value
Error code	-10073
Parameter 1	The table name and script name.

Probable cause A failure occurred when the MobiLink synchronization server was deleting row(s) from the given table in the consolidated database.

Unable to delete user name '%1!s!' from the ml_user_table

Item	Value
Error code	-10083
Parameter 1	The name of the user to be deleted.

Probable cause Please check if the MobiLink user is valid and is not currently in use by other threads.

Unable to determine the remote user name

Item	Value
Error code	-10009

Probable cause The user name that the client sent to the MobiLink synchronization server is invalid. Please make sure that the name is not longer than 128 bytes.

Unable to determine the remote user password

Item	Value
Error code	-10010

Probable cause The MobiLink synchronization server was not able to read the remote user password from the stream.

Unable to determine the remote version

Item	Value
Error code	-10011

Probable cause The MobiLink synchronization server was not able to read the user

synchronization version from the stream.

Unable to execute script ‘%1!s!’

Item	Value
Error code	–10080
Parameter 1	The name of the script.

Probable cause Please check that the SQL statements in the script are valid.

Unable to fetch from table ‘%1!s!’ using %2!s!

Item	Value
Error code	–10071
Parameter 1	The table name and script name.

Probable cause A failure occurred when the MobiLink synchronization server was retrieving row(s) from the given table in the consolidated database.

Unable to flush scripts

Item	Value
Error code	–10020

Probable cause The MobiLink synchronization server always updates its cached scripts at the beginning of each synchronization by connecting to the consolidated database server and querying the ml_scripts_modified table. The MobiLink synchronization server encountered a problem flushing the scripts.

Unable to generate scripts for version ‘%1!s!’

Item	Value
Error code	–10027
Parameter 1	User version string.

Probable cause The MobiLink synchronization server was not able to generate example scripts.

Unable to initialize ODBC

Item	Value
Error code	–10066

Probable cause The MobiLink synchronization server was not able to initialize the ODBC layer. Please make sure that there is enough memory to start the MobiLink synchronization server and then start the MobiLink synchronization server with the recommended ODBC drivers.

Unable to initialize authentication subsystem

Item	Value
Error code	–10068

Probable cause The MobiLink synchronization server was not able to initialize the user authentication layer. Your system may be running out of memory. You may need to close some applications or add more memory to your system.

Unable to initialize consolidated database interface

Item	Value
Error code	–10069

Probable cause The MobiLink synchronization server was not able to initialize the consolidated database interface layer. Please make sure that there is enough memory to start the MobiLink synchronization server and then start the MobiLink synchronization server with the recommended ODBC drivers.

Unable to initialize the resource DLL '%1!s!'

Item	Value
Error code	–10082
Parameter 1	A string that gives the name of the resource DLL.

Probable cause Please check your MobiLink synchronization server installation.

Unable to insert into table ‘%1!s!’ using %2!s!

Item	Value
Error code	–10072
Parameter 1	The table name and script name.

Probable cause A failure occurred when the MobiLink synchronization server was inserting row(s) into the given table in the consolidated database.

Unable to insert user name ‘%1!s!’ into the ml_user table

Item	Value
Error code	–10054
Parameter 1	The user name.

Probable cause The MobiLink synchronization server was not able to add the given user into the ml_user table. Please verify that the consolidated database server is running and that the MobiLink user has permission to modify the ml_user table.

Unable to launch the command: (%1!s!). The system error code is %2!d!

Item	Value
Error code	–10037
Parameter 1	Command and system return code.

Probable cause The MobiLink synchronization server allows users to launch something between upload and download. However, the launch was unsuccessful.

Unable to load UNILIB collation %1!d!: error %2!d!

Item	Value
Error code	–10007
Parameter 1	Unilib charset ID and Unilib function return code.

Probable cause The MobiLink synchronization server cannot initialize the Unilib converter.

Unable to load UNILIB collation expansion factor: error %1!d!

Item	Value
Error code	-10008
Parameter 1	Unilib function return code.

Probable cause The MobiLink synchronization server cannot determine the character expansion factor from the Unilib converter.

Unable to load entry points from dll: '%1!s!'

Item	Value
Error code	-10088
Parameter 1	The DLL name.

Probable cause Please check your MobiLink synchronization server installation.

Unable to open %1!s!

Item	Value
Error code	-10058
Parameter 1	The script name.

Probable cause The MobiLink synchronization server failed to execute the given table script. Please make sure that the script contains valid SQL.

Unable to retry the current transaction after deadlock in the consolidated database. The retry limit has been reached

Item	Value
Error code	-10021

Probable cause The MobiLink synchronization server has retried the current transaction, but the deadlock problem still occurred. Please redesign your synchronization logic or use the -r MobiLink synchronization server command line option.

Unable to update table '%1!s!' using %2!s!

Item	Value
Error code	-10074
Parameter 1	The table name and script name.

Probable cause A failure occurred when the MobiLink synchronization server was updating row(s) for the given table in the consolidated database.

Unrecognized domain id %1!d!

Item	Value
Error code	-10024
Parameter 1	Domain ID.

Probable cause The client-requested domain ID (datatype) is not supported. Please make sure that your MobiLink synchronization server is up-to-date.

Upload failed with client error %1!d!

Item	Value
Error code	-10032
Parameter 1	An error number sent by the client.

Probable cause The MobiLink synchronization server aborted the synchronization because the client indicates there is a problem on the remote site during upload.

User name '%1!s!' not found in the ml_user table

Item	Value
Error code	-10056
Parameter 1	The user name.

Probable cause The MobiLink synchronization server could not find the user from the ml_user table. Please add this user to the ml_user table using dbmluser or start the MobiLink synchronization server with the -zu+ option.

Version '%1!s!' not found in the ml_script_version table. Cannot synchronize

Item	Value
Error code	-10013
Parameter 1	Version string.

Probable cause The client-specified version does not exist in the consolidated database.
Please create a script version.

Version mismatch with dll : '%1!s!'\nExpected version: %2!d! got version: %3!d!

Item	Value
Error code	-10087
Parameter 1	The DLL name, expected version number and the actual version number.

Probable cause Please check your MobiLink synchronization server installation.

CHAPTER 14

MobiLink Synchronization Server Warning Messages

About this chapter

This chapter lists MobiLink synchronization server warnings, as well as their probable causes.

The warning messages are written to the MobiLink synchronization server message log.

Levels can be 1 - 5. The following table explains each level:

Level	Description
1	Server or high level ODBC warnings when a MobiLink synchronization server starts or shuts down
2	Synchronization and user level warnings when a synchronization starts and ends
3	Schema level (including publications and tables) warnings
4	Script and lower level ODBC warnings
5	Table and row level warnings

Contents

Topic:	page
MobiLink synchronization server warning messages sorted by code	438
MobiLink synchronization server warning messages sorted by message	443
MobiLink synchronization server warning descriptions	448

MobiLink synchronization server warning messages sorted by code

Warning code	Level	Warning message
10001	1	“Maximum number of database connections set to %1!lu! (must be at least the number of worker threads plus one)” on page 453
10002	1	“If needed, ODBC cursors will be used, via the Microsoft ODBC Cursor Library, to simulate SQLSETPOS for inserts, updates, and deletes” on page 451
10003	1	“ODBC Isolation level (%1!s!) is not supported” on page 454
10004	1	“ODBC function %1!s! is not supported by the driver” on page 454
10005	1	“ODBC statement option %1!s! has changed from %2!s! (%3!lu!) to %4!s! (%5!lu!)” on page 455
10006	1	“ODBC statement option %1!s! has changed from %2!lu! to %3!lu!” on page 455
10007	2	“Retrying the begin_connection transaction after deadlock in the consolidated database” on page 456
10008	2	“Retry on deadlock is disabled. The MobiLink synchronization server is using an internal workaround which requires this setting” on page 456
10009	2	“MobiLink table ‘%1!s!’ is damaged” on page 453
10010	2	“No handle_error script is defined. The default action code (%1!ld!) will decide the error behavior” on page 454
10011	2	“Unrecognized value (%1!ld!) in ml_user.-commit_state. The state information for this user is probably corrupted” on page 467

Warning code	Level	Warning message
10012	2	“The consolidated and remote databases disagree on when the last synchronization took place. The remote is being asked to send a new upload that starts at the last known synchronization point” on page 460
10013	4	“Expecting %1!d! parameter(s) in cursor, but found %2!d!” on page 450
10014	4	“Expecting at most %1!d! parameter(s) in cursor, but found %2!d!” on page 451
10015	3	“Table ‘%1!s!’ has at least one timestamp column. Due to a timestamp precision mismatch, uploaded timestamps can lose precision, defeating download filtering” on page 458
10016	3	“Table ‘%1!s!’ has at least one timestamp column. Due to a timestamp precision mismatch, downloaded timestamps can lose precision, resulting in inconsistent data” on page 458
10017	3	“The consolidated and remote databases have different timestamp precisions. Consolidated database timestamps are precise to %1!d! digit(s) in the fractional second while the remote database timestamps are precise to %2!d! digit(s)” on page 461
10018	3	“The timestamp precision mismatch may be resolved by setting the DEFAULT_-TIMESTAMP_INCREMENT option on the remote database to %1!d! and TRUNCATE_-TIMESTAMP_VALUES to ‘On’” on page 463
10019	3	“The remote database is not capable of matching the timestamp precision of the consolidated database. Your application, schema, and scripts must contain logic that copes with the precision mismatch” on page 462
10020	3	“The timestamp precision mismatch may affect upload conflict detection. Use the -zp option to cause the MobiLink synchronization server to use the lowest timestamp precision for conflict detection purposes” on page 462

Warning code	Level	Warning message
10021	3	“The remote and consolidated databases have different timestamp precisions, and a timestamp value with a precision higher than the lower-precision side was used for conflict detection purposes. Consider using the -zp option” on page 461
10022	3	“Publication ‘%1!s!’ is not referenced by any table” on page 455
10023	3	“The upload will be rolled back and the synchronization aborted. The next time this remote synchronizes, it will ask what happened to the previous upload” on page 464
10024	3	“Table ‘%1!s!’ has no entry in the %2!s! table” on page 459
10025	5	“Invalid character data encountered in upload – substituting ‘?’” on page 452
10026	5	“Invalid character data encountered in upload – using NULL” on page 452
10027	5	“Invalid character data encountered in upload – using empty string” on page 452
10028	5	“Multi-byte characters truncated on upload” on page 453
10029	5	“Unable to convert character data for download – substituting ‘?’” on page 465
10030	5	“Unable to convert character data for download – using NULL” on page 465
10031	5	“Unable to convert character data for download – using empty string” on page 466
10032	2	“Unable to open the file to store the client synchronization logs. The filename is ‘%1!s!’” on page 466
10033	2	“An error occurred reading the remote client’s synchronization log” on page 449
10034	2	“Unable to write to the local file that contains remote synchronization logs” on page 466

Warning code	Level	Warning message
10035	2	“The remote client’s synchronization log ended prematurely, and was probably truncated” on page 462
10036	2	“Client synchronization logs will be shown in the MobiLink synchronization server output file or the console” on page 450
10037	5	“Ignoring updated row (new values)” on page 451
10038	5	“Ignoring updated row (old values)” on page 452
10039	2	“Error detected while using multi-row cursor – retrying with single row cursor” on page 450
10040	2	“%1!lu! row(s) were ignored in updating table %2!s!” on page 448
10041	2	“The upload will be committed and the synchronization aborted. The next time this remote synchronizes, it will ask what happened to the previous upload” on page 463
10042	1	“NT Performance Monitor data area failed to initialize” on page 453
10043	4	“Unable to determine current timestamp from consolidated database” on page 466
10044	5	“A row in table ‘%1!s!’ could not be updated because it no longer exists in the consolidated database” on page 449
10045	2	“Retrying the upload after deadlock in the consolidated database” on page 457
10046	2	“Retrying the upload. Working around a known ODBC driver problem” on page 457
10047	4	“Cannot directly determine the name of the table referenced by the cursor. The table name is required for inserts, updates, and deletes when using the Microsoft ODBC Cursor Library” on page 449

Warning code	Level	Warning message
10048	2	“Retrying the begin_synchronization transaction after deadlock in the consolidated database” on page 456
10049	2	“Retrying the end_synchronization transaction after deadlock in the consolidated database” on page 457
10050	4	“%1!s!” on page 448
10051	1	“Unrecognized ODBC driver ‘%1!s!’. The functionality and quality of ODBC drivers varies greatly. This driver may lack functionality required for successful synchronizations. Use at your own risk” on page 467
10052	1	“The upload_cursor, new_row_cursor, and old_row_cursor scripts are deprecated. It is strongly recommended that you use the statement-based upload scripts instead” on page 464
10053	1	“The -zac switch is deprecated. It is strongly recommended that you use the -za switch instead” on page 459
10054	1	“The -zec switch is deprecated. It is strongly recommended that you use the -ze switch instead” on page 459
10055	2	“The client has provided %1!d! authentication parameter(s), but no authenticate_parameters script exists” on page 460
10056	4	“There is no download data script defined for table: %1!s!. Synchronization has the risk of potentially losing download data” on page 464
10057	4	“There is no upload data script defined for table: %1!s!. Synchronization has the risk of potentially losing upload data” on page 465

MobiLink synchronization server warning messages sorted by message

Warning code	Level	Warning message
10040	2	“%1!lu! row(s) were ignored in updating table %2!s!” on page 448
10050	4	“%1!s!” on page 448
10044	5	“A row in table ‘%1!s!’ could not be updated because it no longer exists in the consolidated database” on page 449
10033	2	“An error occurred reading the remote client’s synchronization log” on page 449
10047	4	“Cannot directly determine the name of the table referenced by the cursor. The table name is required for inserts, updates, and deletes when using the Microsoft ODBC Cursor Library” on page 449
10036	2	“Client synchronization logs will be shown in the MobiLink synchronization server output file or the console” on page 450
10039	2	“Error detected while using multi-row cursor – retrying with single row cursor” on page 450
10013	4	“Expecting %1!ld! parameter(s) in cursor, but found %2!ld!” on page 450
10014	4	“Expecting at most %1!ld! parameter(s) in cursor, but found %2!ld!” on page 451
10002	1	“If needed, ODBC cursors will be used, via the Microsoft ODBC Cursor Library, to simulate SQLSETPOS for inserts, updates, and deletes” on page 451
10037	5	“Ignoring updated row (new values)” on page 451
10038	5	“Ignoring updated row (old values)” on page 452
10025	5	“Invalid character data encountered in upload – substituting ‘?’” on page 452

Warning code	Level	Warning message
10026	5	“Invalid character data encountered in upload – using NULL” on page 452
10027	5	“Invalid character data encountered in upload – using empty string” on page 452
10001	1	“Maximum number of database connections set to %1!lu! (must be at least the number of worker threads plus one)” on page 453
10009	2	“MobiLink table ‘%1!s!’ is damaged” on page 453
10028	5	“Multi-byte characters truncated on upload” on page 453
10042	1	“NT Performance Monitor data area failed to initialize” on page 453
10010	2	“No handle_error script is defined. The default action code (%1!d!) will decide the error behavior” on page 454
10003	1	“ODBC Isolation level (%1!s!) is not supported” on page 454
10004	1	“ODBC function %1!s! is not supported by the driver” on page 454
10006	1	“ODBC statement option %1!s! has changed from %2!lu! to %3!lu!” on page 455
10005	1	“ODBC statement option %1!s! has changed from %2!s! (%3!lu!) to %4!s! (%5!lu!)” on page 455
10022	3	“Publication ‘%1!s!’ is not referenced by any table” on page 455
10008	2	“Retry on deadlock is disabled. The MobiLink synchronization server is using an internal workaround which requires this setting” on page 456
10007	2	“Retrying the begin_connection transaction after deadlock in the consolidated database” on page 456

Warning code	Level	Warning message
10048	2	“Retrying the begin_synchronization transaction after deadlock in the consolidated database” on page 456
10049	2	“Retrying the end_synchronization transaction after deadlock in the consolidated database” on page 457
10045	2	“Retrying the upload after deadlock in the consolidated database” on page 457
10046	2	“Retrying the upload. Working around a known ODBC driver problem” on page 457
10016	3	“Table ‘%1!s!’ has at least one timestamp column. Due to a timestamp precision mismatch, downloaded timestamps can lose precision, resulting in inconsistent data” on page 458
10015	3	“Table ‘%1!s!’ has at least one timestamp column. Due to a timestamp precision mismatch, uploaded timestamps can lose precision, defeating download filtering” on page 458
10024	3	“Table ‘%1!s!’ has no entry in the %2!s! table” on page 459
10053	1	“The -zac switch is deprecated. It is strongly recommended that you use the -za switch instead” on page 459
10054	1	“The -zec switch is deprecated. It is strongly recommended that you use the -ze switch instead” on page 459
10055	2	“The client has provided %1!d! authentication parameter(s), but no authenticate_parameters script exists” on page 460
10012	2	“The consolidated and remote databases disagree on when the last synchronization took place. The remote is being asked to send a new upload that starts at the last known synchronization point” on page 460

Warning code	Level	Warning message
10017	3	“The consolidated and remote databases have different timestamp precisions. Consolidated database timestamps are precise to %!d! digit(s) in the fractional second while the remote database timestamps are precise to %2!d! digit(s)” on page 461
10021	3	“The remote and consolidated databases have different timestamp precisions, and a timestamp value with a precision higher than the lower-precision side was used for conflict detection purposes. Consider using the -zp option” on page 461
10035	2	“The remote client’s synchronization log ended prematurely, and was probably truncated” on page 462
10019	3	“The remote database is not capable of matching the timestamp precision of the consolidated database. Your application, schema, and scripts must contain logic that copes with the precision mismatch” on page 462
10020	3	“The timestamp precision mismatch may affect upload conflict detection. Use the -zp option to cause the MobiLink synchronization server to use the lowest timestamp precision for conflict detection purposes” on page 462
10018	3	“The timestamp precision mismatch may be resolved by setting the DEFAULT_TIMESTAMP_INCREMENT option on the remote database to %!d! and TRUNCATE_TIMESTAMP_VALUES to ‘On’” on page 463
10041	2	“The upload will be committed and the synchronization aborted. The next time this remote synchronizes, it will ask what happened to the previous upload” on page 463
10023	3	“The upload will be rolled back and the synchronization aborted. The next time this remote synchronizes, it will ask what happened to the previous upload” on page 464

Warning code	Level	Warning message
10052	1	“The upload_cursor, new_row_cursor, and old_row_cursor scripts are deprecated. It is strongly recommended that you use the statement-based upload scripts instead” on page 464
10056	4	“There is no download data script defined for table: %!s!. Synchronization has the risk of potentially losing download data” on page 464
10057	4	“There is no upload data script defined for table: %!s!. Synchronization has the risk of potentially losing upload data” on page 465
10029	5	“Unable to convert character data for download – substituting ‘?’” on page 465
10030	5	“Unable to convert character data for download – using NULL” on page 465
10031	5	“Unable to convert character data for download – using empty string” on page 466
10043	4	“Unable to determine current timestamp from consolidated database” on page 466
10032	2	“Unable to open the file to store the client synchronization logs. The filename is ‘%!s!’” on page 466
10034	2	“Unable to write to the local file that contains remote synchronization logs” on page 466
10051	1	“Unrecognized ODBC driver ‘%!s!’. The functionality and quality of ODBC drivers varies greatly. This driver may lack functionality required for successful synchronizations. Use at your own risk” on page 467
10011	2	“Unrecognized value (%!ld!) in ml_user-commit_state. The state information for this user is probably corrupted” on page 467

MobiLink synchronization server warning descriptions

This section provides a full listing of warning messages and descriptions.

Warnings with an ODBC state marked “handled by ODBC driver” are not returned to ODBC applications, as the ODBC driver carries out the required actions.

%1!lu! row(s) were ignored in updating table %2!s!

Item	Value
Warning code	10040
Level	2
Parameter 1	The total number of ignored rows and the name of the table.

Probable cause The MobiLink synchronization server counts all the upload rows that were not applied to the consolidated database as ignored rows. This can happen if there are upload inserts from the client, but there is no upload_insert script for the listed table in the consolidated database.

%1!s!

Item	Value
Warning code	10050
Level	4
Parameter 1	A message from the ODBC driver.

Probable cause The MobiLink synchronization server made a successful ODBC call, but the ODBC driver displayed a warning message.
Avoiding this message depends on the specific warning.

A row in table ‘%1!s!’ could not be updated because it no longer exists in the consolidated database

Item	Value
Warning code	10044
Level	5
Parameter 1	Table name.

Probable cause An update statement failed because the table in the consolidated database doesn't contain the original row.

An error occurred reading the remote client's synchronization log

Item	Value
Warning code	10033
Level	2

Probable cause The MobiLink synchronization server was unable to get the remote error log from the client. To avoid this warning, please do not kill the client when it is running and also make sure the network connection is okay.

Cannot directly determine the name of the table referenced by the cursor. The table name is required for inserts, updates, and deletes when using the Microsoft ODBC Cursor Library

Item	Value
Warning code	10047
Level	4

Probable cause The MobiLink synchronization server was not able to find the table name referenced by the cursor. To avoid this warning, please use statement-based synchronization.

Client synchronization logs will be shown in the MobiLink synchronization server output file or the console

Item	Value
Warning code	10036
Level	2

Probable cause If an error occurs on the client side during synchronization, the client may send its output file to the server and the server will store this output file to a file specified using the server switch -e or -et. However, if the MobiLink synchronization server could not open this file for writing, it will show this warning message and will write the remote log into its output file or console.

To avoid this warning, please make sure the MobiLink synchronization server has the privilege to write the file.

Error detected while using multi-row cursor – retrying with single row cursor

Item	Value
Warning code	10039
Level	2

Probable cause Errors were detected when the MobiLink synchronization server applied the upload stream using multi-row cursors (bulk operation). It will roll back the upload stream and retry the upload transaction using single-row cursors.

Expecting %1!ld! parameter(s) in cursor, but found %2!ld!

Item	Value
Warning code	10013
Level	4
Parameter 1	The number of parameter(s) expected and the number of parameter(s) found.

Probable cause The number of parameters is not as expected. Check the script to ensure it is correct.

Expecting at most %1!d! parameter(s) in cursor, but found %2!d!

Item	Value
Warning code	10014
Level	4
Parameter 1	The maximum number of parameter(s) expected and the number of parameter(s) found.

Probable cause There is a maximum number of parameters for every cursor script in the MobiLink synchronization server. If the number of parameters for the given cursor script is larger than the maximum number of parameters, the server will show this warning.

If needed, ODBC cursors will be used, via the Microsoft ODBC Cursor Library, to simulate SQLSETPOS for inserts, updates, and deletes

Item	Value
Warning code	10002
Level	1

Probable cause The MobiLink synchronization server requires some functionality not provided by your selected ODBC driver. Updating to a newer driver may resolve this problem.

Ignoring updated row (new values)

Item	Value
Warning code	10037
Level	5

Probable cause There is a conflict-update, but there is no `upload_new_row_insert` or `new_row_cursor` script defined in the consolidated database for the table.

Ignoring updated row (old values)

Item	Value
Warning code	10038
Level	5

Probable cause There is a conflict-update, but there is no `upload_old_row_insert` or `old_row_cursor` script defined in the consolidated database for the table.

Invalid character data encountered in upload – substituting ‘?’

Item	Value
Warning code	10025
Level	5

Probable cause Character data originating from the client needs to be translated before being entered into the consolidated database. This was not possible for all characters. A ‘?’ will be substituted for each problematic character.

Invalid character data encountered in upload – using NULL

Item	Value
Warning code	10026
Level	5

Probable cause Character data originating from the client needs to be translated before being entered into the consolidated database. This was not possible. NULL will be entered instead.

Invalid character data encountered in upload – using empty string

Item	Value
Warning code	10027
Level	5

Probable cause Character data originating from the client needs to be translated before being entered into the consolidated database. This was not possible, an empty string will be entered instead.

Maximum number of database connections set to %1!lu! (must be at least the number of worker threads plus one)

Item	Value
Warning code	10001
Level	1
Parameter 1	Maximum number of connections.

Probable cause The MobiLink synchronization server makes one connection for each worker thread and an extra connection for the main thread. Therefore, the maximum number of connections must be at least the number of worker threads plus one.

MobiLink table '%1!s!' is damaged

Item	Value
Warning code	10009
Level	2
Parameter 1	The MobiLink system table name.

Probable cause The MobiLink synchronization server was unable to get information from the listed table. Please make sure the table does exist and the database server is running.

Multi-byte characters truncated on upload

Item	Value
Warning code	10028
Level	5

Probable cause This is an internal error, and should never be reported.

NT Performance Monitor data area failed to initialize

Item	Value
Warning code	10042
Level	1

Probable cause The NT Performance Monitor will not be able to monitor this instance of the MobiLink synchronization server. Only one instance of the MobiLink synchronization server may be monitored at a time. The instance that may be monitored is always the one running the instance that was started at the earliest time.

Avoid this warning by making sure the MobiLink synchronization server you want to use with the Performance Monitor is started before any other MobiLink synchronization server on the same machine.

No handle_error script is defined. The default action code (%1!ld!) will decide the error behavior

Item	Value
Warning code	10010
Level	2
Parameter 1	The error action code.

Probable cause An error occurred in the MobiLink synchronization server during synchronization. However there is no handle_error script defined in the consolidated database. The server will take the default action for the error. To avoid this warning, please define a handle_error script.

ODBC Isolation level (%1!s!) is not supported

Item	Value
Warning code	10003
Level	1
Parameter 1	The required isolation level.

Probable cause The required isolation level is not supported by the consolidated database. Determine if another level is suitable.

ODBC function %1!s! is not supported by the driver

Item	Value
Warning code	10004
Level	1
Parameter 1	ODBC function name.

Probable cause This function is required for the MobiLink synchronization server to operate. Update your ODBC driver.

ODBC statement option %1!s! has changed from %2!lu! to %3!lu!

Item	Value
Warning code	10006
Level	1

Probable cause The option has been changed by the ODBC driver. This may not be desirable.

ODBC statement option %1!s! has changed from %2!s! (%3!lu!) to %4!s! (%5!lu!)

Item	Value
Warning code	10005
Level	1

Probable cause The option has been changed by the ODBC driver. This may not be desirable.

Publication '%1!s!' is not referenced by any table

Item	Value
Warning code	10022
Level	3
Parameter 1	Publication name.

Probable cause The MobiLink synchronization client sends an upload stream that includes upload data as well as upload tables, publications, etc. All these publications must be referenced by at least one of the upload tables. If there are any publications that are not referenced by any upload table, the server will show this warning. If this happens, please contact technical support.

Retry on deadlock is disabled. The MobiLink synchronization server is using an internal workaround which requires this setting

Item	Value
Warning code	10008
Level	2

Probable cause The MobiLink synchronization server will not retry a synchronization when deadlock occurs. This warning should only occur when using the Oracle 8.0.5.7 ODBC driver (which is not recommended).

Retrying the begin_connection transaction after deadlock in the consolidated database

Item	Value
Warning code	10007
Level	2

Probable cause Deadlock occurred in the transaction of begin_transaction in the consolidated database. To avoid this warning, please rewrite your begin_connection script to avoid deadlocks.

Retrying the begin_synchronization transaction after deadlock in the consolidated database

Item	Value
Warning code	10048
Level	2

Probable cause Deadlock occurred when the MobiLink synchronization server executed the begin_synchronization script. It will roll back the transaction and retry this script.

Retrying the end_synchronization transaction after deadlock in the consolidated database

Item	Value
Warning code	10049
Level	2

Probable cause Deadlock occurred when the MobiLink synchronization server executed the end_synchronization script. It will roll back the transaction and retry this script.

Retrying the upload after deadlock in the consolidated database

Item	Value
Warning code	10045
Level	2

Probable cause Deadlock occurred when the MobiLink synchronization server was applying the upload stream. It will roll back the transaction and retry this script.

Avoid this warning by removing contention between synchronization scripts. There can also be contention between synchronization scripts and other applications.

Retrying the upload. Working around a known ODBC driver problem

Item	Value
Warning code	10046
Level	2

Probable cause A quirk in the ODBC driver requires the MobiLink synchronization server to retry the upload; otherwise it will not be applied successfully. This warning should only occur when using the Oracle 8.0.5.7 ODBC driver, which is not recommended.

Table ‘%1!s!’ has at least one timestamp column. Due to a timestamp precision mismatch, downloaded timestamps can lose precision, resulting in inconsistent data

Item	Value
Warning code	10016
Level	3
Parameter 1	Table name.

Probable cause A remote database is synchronizing a table with at least one timestamp column while the timestamp precision of the remote database is lower than that of the consolidated database. The downloaded value will lose precision after being stored into the remote database.

 This situation creates a virtual difference in the synchronized timestamp data between the consolidated database and the remote database. To avoid this data inconsistency problem, you should align timestamp precision across all databases involved in your synchronization system.

Table ‘%1!s!’ has at least one timestamp column. Due to a timestamp precision mismatch, uploaded timestamps can lose precision, defeating download filtering

Item	Value
Warning code	10015
Level	3
Parameter 1	Table name.

Probable cause A remote database is synchronizing a table with at least one timestamp column while the timestamp precision on the remote database is higher than that of the consolidated database. The uploaded value will lose precision after being stored in the consolidated database. The MobiLink synchronization server compares upload rows and download rows in order to filter away redundant download rows.

 The loss of precision may create a virtual difference that defeats the download filtering. This situation can affect download performance. To avoid this performance penalty, you should align timestamp precision across all databases involved in your synchronization system.

Table '%1!s!' has no entry in the %2!s! table

Item	Value
Warning code	10024
Level	3
Parameter 1	Table name.

Probable cause The tables ml_table, ml_table_script, ml_script contain the scripts for every table that participates in synchronization. The MobiLink synchronization server will show this warning for all the synchronization tables that have no script or that have no entry in the table ml_table.

The -zac switch is deprecated. It is strongly recommended that you use the -za switch instead

Item	Value
Warning code	10053
Level	1

Probable cause The cursor-based upload scripts will not be supported in future releases of the MobiLink synchronization server.

The -zec switch is deprecated. It is strongly recommended that you use the -ze switch instead

Item	Value
Warning code	10054
Level	1

Probable cause The cursor-based upload scripts will not be supported in future releases of the MobiLink synchronization server.

The client has provided %1!d! authentication parameter(s), but no authenticate_parameters script exists

Item	Value
Warning code	10055
Level	2
Parameter 1	The number of authentication parameter(s).

Probable cause

The MobiLink synchronization client sent the listed number of authentication parameters. However, there is no authentication parameter script defined in the consolidated database.

To avoid this warning, please define the `authenticate_parameters` script in the consolidated database or do not send authentication parameter(s) from the client.

The consolidated and remote databases disagree on when the last synchronization took place. The remote is being asked to send a new upload that starts at the last known synchronization point

Item	Value
Warning code	10012
Level	2

Probable cause

The MobiLink synchronization server stores the remote ending log offsets in its system tables after every synchronization and it compares these ending log offsets with the remote beginning offsets when a new synchronization comes in. If the ending offsets do not match the beginning offsets, the server will show this warning and also inform the client about the mismatching offsets.

The consolidated and remote databases have different timestamp precisions. Consolidated database timestamps are precise to %1!d! digit(s) in the fractional second while the remote database timestamps are precise to %2!d! digit(s)

Item	Value
Warning code	10017
Level	3
Parameter 1	Timestamp precision of the consolidated database in terms of number of decimal digits in the fractional second.
Parameter 2	Timestamp precision of the remote database in terms of number of decimal digits in the fractional second.

Probable cause Inconsistent timestamp precisions were found between the remote database and the consolidated database. Align the databases to the same precision to avoid a performance penalty or inconsistent data.

The remote and consolidated databases have different timestamp precisions, and a timestamp value with a precision higher than the lower-precision side was used for conflict detection purposes. Consider using the -zp option

Item	Value
Warning code	10021
Level	3

Probable cause An upload conflict is detected based on a tolerable timestamp difference while the -zp switch is not used. If you decided not to align precision on the databases involved in your synchronization system, you may use the -zp switch to start the MobiLink synchronization server.

If the switch is used, MobiLink synchronization server will tolerate conflict caused by timestamp differences smaller than the lower precision among the two databases.

The remote client's synchronization log ended prematurely, and was probably truncated

Item	Value
Warning code	10035
Level	2

Probable cause The MobiLink synchronization server was not able to completely get the remote error log from the client, so the remote error log may have been truncated. To avoid this warning, please do not kill the client when it is running and also make sure the network connection is okay.

The remote database is not capable of matching the timestamp precision of the consolidated database. Your application, schema, and scripts must contain logic that copes with the precision mismatch

Item	Value
Warning code	10019
Level	3

Probable cause Timestamp precision of the consolidated database is found to be higher than attainable by the remote database. If possible you may lower the timestamp precision on the consolidated database in order to avoid inconsistent timestamp data between the remote and the consolidated database.

Otherwise, you may need to avoid synchronizing timestamps in your synchronization schema; or you may need to have conflict detection scripts aware of the virtual difference; or you may need to use the `-zp` switch to tolerate the conflict. Your application should also be able to deal with the inconsistency.

The timestamp precision mismatch may affect upload conflict detection. Use the `-zp` option to cause the MobiLink synchronization server to use the lowest timestamp precision for conflict detection purposes

Item	Value
Warning code	10020
Level	3

Probable cause

Timestamp precision mismatch between the remote database and the consolidated database has been detected. The mismatch can affect upload conflict detection as the MobiLink synchronization server will compare the rows for the two databases. If you decided not to align precision on the databases involved in your synchronization system, you may use the `-zp` switch to start the MobiLink synchronization server.

If this switch is used, the MobiLink synchronization server will tolerate conflict caused by timestamp differences smaller than the lower precision among the two databases.

The timestamp precision mismatch may be resolved by setting the `DEFAULT_TIMESTAMP_INCREMENT` option on the remote database to `%1!d!` and `TRUNCATE_TIMESTAMP_VALUES` to 'On'

Item	Value
Warning code	10018
Level	3
Parameter 1	Timestamp precision of the consolidated database in terms of number of decimal digits in the fractional second.

Probable cause

This is an advisory on how to align timestamp precision by adjusting timestamp precision on the ASA client database or the ASA reference database for UltraLite clients. UltraLite clients need to be regenerated after the precision is adjusted.

The upload will be committed and the synchronization aborted. The next time this remote synchronizes, it will ask what happened to the previous upload

Item	Value
Warning code	10041
Level	2

Probable cause

This is an internal warning that is primarily used for testing, but may also be seen as part of technical support engagements.

The upload will be rolled back and the synchronization aborted. The next time this remote synchronizes, it will ask what happened to the previous upload

Item	Value
Warning code	10023
Level	3

Probable cause This is an internal warning that is primarily used for testing, but may also be seen as part of technical support engagements.

The upload_cursor, new_row_cursor, and old_row_cursor scripts are deprecated. It is strongly recommended that you use the statement-based upload scripts instead

Item	Value
Warning code	10052
Level	1

Probable cause The cursor-based upload scripts will not be supported in future releases of the MobiLink synchronization server. To avoid this warning, please convert all your cursor-based upload scripts into statement-based upload scripts.

There is no download data script defined for table: %1!s!. Synchronization has the risk of potentially losing download data

Item	Value
Warning code	10056
Level	4
Parameter 1	Table name.

Probable cause The listed table is involved in a download-only synchronization. However, there is no download script for this table in the consolidated database.

To avoid this warning, please define download script(s) for this table in the consolidated database, or always do full synchronization.

There is no upload data script defined for table: %1!s!. Synchronization has the risk of potentially losing upload data

Item	Value
Warning code	10057
Level	4
Parameter 1	Table name.

Probable cause

The listed table is involved in the synchronization and there are some changes to this table in the remote data. However, there is no upload script for this table in the dbmlsync consolidated database.

To avoid this warning, please define upload script(s) for this table in the consolidated database, or do not make any changes in the remote database using any other application except the MobiLink synchronization client.

Unable to convert character data for download – substituting ‘?’

Item	Value
Warning code	10029
Level	5

Probable cause

Character data originating from the consolidated database needs to be translated before being sent to the client. This was not possible for all characters. A ‘?’ will be substituted for each problem character.

Unable to convert character data for download – using NULL

Item	Value
Warning code	10030
Level	5

Probable cause

Character data originating from the consolidated database needs to be translated before being sent to the client. This was not possible. NULL will be sent instead.

Unable to convert character data for download – using empty string

Item	Value
Warning code	10031
Level	5

Probable cause Character data originating from the consolidated database needs to be translated before being sent to the client. This was not possible. An empty string will be sent instead.

Unable to determine current timestamp from consolidated database

Item	Value
Warning code	10043
Level	4

Probable cause The MobiLink synchronization server was not able to get the current timestamp from the consolidated database. Please make sure the database server is running and the network connection is okay.

Unable to open the file to store the client synchronization logs. The filename is '%1!s!'

Item	Value
Warning code	10032
Level	2

Probable cause The MobiLink synchronization server was unable to open the local remote log file given by option -e or -et. Please make sure the file name and path are valid and the file is writable.

Unable to write to the local file that contains remote synchronization logs

Item	Value
Warning code	10034
Level	2

Probable cause The MobiLink synchronization server was unable to write the remote error log to a local file. To avoid this warning, please make sure the file name and

path given by option -e or -et are valid and the file is writable.

Unrecognized ODBC driver ‘%1!s!’. The functionality and quality of ODBC drivers varies greatly. This driver may lack functionality required for successful synchronizations. Use at your own risk

Item	Value
Warning code	10051
Level	1
Parameter 1	The file name of an ODBC driver.

Probable cause The MobiLink synchronization server is very well tested with a set of ODBC drivers. However, the ODBC driver you are currently using is not on the list. To avoid this warning, please run the MobiLink synchronization server with a recommended ODBC driver.

Unrecognized value (%1!ld!) in ml_user.commit_state. The state information for this user is probably corrupted

Item	Value
Warning code	10011
Level	2
Parameter 1	The value of a commit state.

Probable cause The MobiLink synchronization server stores the last synchronization status for an UltraLite application in the commit_state column in the ml_user table. However, the server does not recognize the commit state fetched from the consolidated database. Please do not manually modify the values of the commit_state in ml_table.

Index

Symbols

-D option		MobiLink [dbmlsrv9] -sl java	19
-DMLStartClasses		-classpath option	
-MLAutoLoadPath option		MobiLink [dbmlsrv9] -sl java	19
-MLDomConfigFile option		-clrConGC option	
-MLStartClasses		MobiLink [dbmlsrv9] -sl dnet	17
-a option		-clrFlavor option	
MobiLink [dbmlsrv9]	8	MobiLink [dbmlsrv9] -sl dnet	17
MobiLink [dbmlsync]	40	-clrVersion option	
-ac option		MobiLink [dbmlsrv9] -sl dnet	17
MobiLink [mlxtract]	304	-cn option	
-al option		MobiLink [dbmlsrv9]	11
MobiLink [mlxtract]	304	-cp option	
-an option		MobiLink [dbmlsrv9] -sl java	19
MobiLink [mlxtract]	304	-cr option	
-ap option		MobiLink [dbmlsrv9]	11
MobiLink [dbmlsync]	40	-ct option	
-b option		MobiLink [dbmlsrv9]	11
MobiLink [dbmlsrv9]	8	-d option	
-ba option		MobiLink [dbasinst]	300
MobiLink [dbmlsync]	40	MobiLink [dbmlsrv9]	11
-bc option		MobiLink [dbmlsync]	42
MobiLink [dbmlsrv9]	9	MobiLink [dbmluser]	308
MobiLink [dbmlsync]	40	-dl option	
-be option		MobiLink [dbmlsrv9]	12
MobiLink [dbmlsync]	41	MobiLink [dbmlsync]	43
-bg option		MobiLink [dbmluser]	308
MobiLink [dbmlsync]	41	-ds option	
-bn option		MobiLink [dbmlsync]	43
MobiLink [dbmlsrv9]	10	-e CommunicationAddress	
-c option		dbmlsync extended option	45
MobiLink [dbmlsrv9]	10	-e CommunicationType	
MobiLink [dbmlsync]	42	dbmlsync extended option	46
MobiLink [dbmluser]	308	-e ConflictRetries	
MobiLink [gencert]	311	dbmlsync extended option	47
MobiLink [mlxtract]	304	-e DisablePolling	
-classic option		dbmlsync extended option	47
		-e DownloadBufferSize	
		dbmlsync extended option	48
		-e DownloadOnly	
		dbmlsync extended option	49
		-e ErrorLogSendLimit	
		dbmlsync extended option	50

-e FireTriggers		dbmlsync extended option	69
-e HoverRescanThreshold	51	-e VerboseUpload	70
-e IgnoreHookErrors	52	-e adr	45
-e IgnoreScheduling	53	-e cr	47
-e Increment	53	-e ctp	46
-e LockTables	54	-e dbs	48
-e Memory	55	-e dir	58
-e MobiLinkPwd	56	-e ds	49
-e NewMobiLinkPwd	56	-e eh	53
-e OfflineDirectory	57	-e el	50
-e PollingPeriod	58	-e ft	51
-e Schedule	58	-e hrt	52
-e ScriptVersion	59	-e inc	54
-e SendColumnNames	61	-e isc	53
-e SendDownloadACK	62	-e lt	55
-e SendTriggers	62	-e mem	56
-e TableOrder	63	-e mn	57
-e UploadOnly	64	-e mp	56
-e Verbose	65	-e option	12
-e VerboseHooks	65	MobiLink [dbmlsrv9]	44
-e VerboseMin	66	-e p	47
-e VerboseOptions	67	-e pp	58
-e VerboseRowCounts	68	-e sa	62
-e VerboseRowValues	68	-e sch	59
		-e st	

dbmlsync extended option	63	-it option	
-e sv		MobiLink [mlxtract]	304
dbmlsync extended option	61, 62	-j option	
-e tor		MobiLink [mlxtract]	304
dbmlsync extended option	64	-jrepath option	
-e uo		MobiLink [dbmlsrv9] -sl java	19
dbmlsync extended option	65	-k option	
-e v		MobiLink [dbasinst]	300
dbmlsync extended option	65	MobiLink [dbmlsync]	72
-e vm		-l option	
dbmlsync extended option	67	MobiLink [dbmlsync]	72
-e vn		MobiLink [mlxtract]	304
dbmlsync extended option	68	-mn option	
-e vo		MobiLink [dbmlsync]	73
dbmlsync extended option	68	-mp option	
-e vr		MobiLink [dbmlsync]	73
dbmlsync extended option	69	-n option	
-e vs		MobiLink [dbasinst]	300
dbmlsync extended option	66	MobiLink [dbmlsync]	73
-e vu		-o option	
dbmlsync extended option	70	MobiLink [dbmlsrv9]	13
-eh option		MobiLink [dbmlsync]	74
MobiLink [dbmlsync]	71	MobiLink [dbmluser]	308
-ek option		MobiLink [mlxtract]	304
MobiLink [dbmlsync]	71	-on option	
-ep option		MobiLink [dbmlsrv9]	14
MobiLink [dbmlsync]	71	-oq option	
-et option		MobiLink [dbmlsrv9]	15
MobiLink [dbmlsrv9]	12	-os option	
-eu option		MobiLink [dbmlsrv9]	15
MobiLink [dbmlsync]	71	MobiLink [dbmlsync]	74
-f option		MobiLink [dbmluser]	308
MobiLink [dbmlsrv9]	13	-ot option	
MobiLink [dbmlstop]	303	MobiLink [dbmlsrv9]	16
MobiLink [dbmluser]	308	MobiLink [dbmlsync]	75
-fr option		MobiLink [dbmluser]	308
MobiLink [dbmlsrv9]	13	-p option	
-h option		MobiLink [dbmlsync]	75
MobiLink [dbmlstop]	303	MobiLink [dbmluser]	308
-hotspot option		MobiLink [mlxtract]	304
MobiLink [dbmlsrv9] -sl java	19	-pc option	
-i option		MobiLink [dbmluser]	308
MobiLink [dbmlsync]	72	-pd option	
-id option		MobiLink [dbmlsync]	75
MobiLink [mlxtract]	304	-pi option	
-is option		MobiLink [dbmlsync]	76
MobiLink [dbmlsync]	72	-pp option	

MobiLink [dbmlsync]	77	-uo option	
-ps option		MobiLink [dbmlsync]	80
MobiLink [dbmlsrv9]	16	-urc option	
-q option		MobiLink [dbmlsync]	80
MobiLink [dbmlsrv9]	16	-v option	
MobiLink [dbmlstop]	303	MobiLink [dbasinst]	300
MobiLink [dbmluser]	308	MobiLink [dbmlsrv9]	21
MobiLink [gencert]	311	MobiLink [dbmlsync]	80
MobiLink [mlxtract]	304	MobiLink [mlxtract]	304
MobiLink client [dbmlsync]	77	-v+ option	
-r option		MobiLink [dbmlsrv9]	21
MobiLink [dbmlsrv9]	16	MobiLink [dbmlsync]	80
MobiLink [dbmlsync]	77	-vc option	
MobiLink [gencert]	311	MobiLink [dbmlsrv9]	21
MobiLink [mlxtract]	304	MobiLink [dbmlsync]	80
-ra option		-verbose option	
MobiLink [dbmlsync]	77	MobiLink [dbmlsrv9] -sl java	19
-rb option		-vf option	
MobiLink [dbmlsync]	77	MobiLink [dbmlsrv9]	21
-rd option		-vh option	
MobiLink [dbmlsrv9]	17	MobiLink [dbmlsrv9]	21
-s option		-vn option	
MobiLink [dbmlsrv9]	17	MobiLink [dbmlsrv9]	21
MobiLink [gencert]	311	MobiLink [dbmlsync]	80
-s7 option		-vo option	
MobiLink [mlxtract]	304	MobiLink [dbmlsync]	80
-sc option		-vp option	
MobiLink [dbmlsync]	79	MobiLink [dbmlsrv9]	21
-server option		MobiLink [dbmlsync]	80
MobiLink [dbmlsrv9] -sl java	19	-vr option	
-sl dnet option		MobiLink [dbmlsrv9]	21
MobiLink [dbmlsrv9]	17	MobiLink [dbmlsync]	80
-sl java option		-vs option	
MobiLink [dbmlsrv9]	19	MobiLink [dbmlsrv9]	21
-t option		MobiLink [dbmlsync]	80
MobiLink [dbmlsrv9]	19	-vt option	
MobiLink [dbmlstop]	303	MobiLink [dbmlsrv9]	21
-tt option		-vu option	
MobiLink [dbmlsrv9]	20	MobiLink [dbmlsrv9]	21
-u option		MobiLink [dbmlsync]	80
MobiLink [dbasinst]	300	-w option	
MobiLink [dbmlsrv9]	20	MobiLink [dbmlsrv9]	22
MobiLink [dbmlsync]	79	MobiLink [dbmlstop]	303
MobiLink [dbmluser]	308	-wc option	
MobiLink [mlxtract]	304	MobiLink [dbmlsync]	81
-ud option		-wu option	
MobiLink [dbmlsrv9]	20	MobiLink [dbmlsrv9]	23

-x option		dbmsync	36
MobiLink [dbmlsrv9]	24	Adaptive Server Enterprise	
MobiLink [dbmlsrv9] -sl java	19	begin_connection_autocommit event	
MobiLink [dbmsync]	82	109	
MobiLink [mlxtract]	304	conversion of data types in MobiLink	
-xf option		synchronization	324
MobiLink [mlxtract]	304	using DDL in MobiLink	109
-xh option		adding	
MobiLink [mlxtract]	304	elliptic-curve and RSA certificates	311
-xp option		MobiLink .NET connection scripts	264
MobiLink [mlxtract]	304	MobiLink .NET table scripts	265
-y option		MobiLink Java connection scripts	266
MobiLink [mlxtract]	304	MobiLink Java table scripts	267
-za option		MobiLink SQL connection scripts	262
MobiLink [dbmlsrv9]	28	MobiLink SQL table scripts	263
-ze option		user names in MobiLink	308
MobiLink [dbmlsrv9]	29	ADDRESS clause	
-zp option		CREATE SYNCHRONIZATION	
MobiLink [dbmlsrv9]	30	USER	245
-zs option		adr	
MobiLink [dbmlsrv9]	30	dbmsync extended option	45
-zt option		ALTER PUBLICATION statement	
MobiLink [dbmlsrv9]	31	SQL syntax	234
-zu option		ALTER SYNCHRONIZATION	
MobiLink [dbmlsrv9]	31	SUBSCRIPTION statement	
-zw option		SQL syntax	236
MobiLink [dbmlsrv9]	31	ALTER SYNCHRONIZATION USER	
-zwd option		statement	
MobiLink [dbmlsrv9]	32	SQL syntax	238
-zwe option		altering	
MobiLink [dbmlsrv9]	33	publications	234
.NET CLR		applications	
MobiLink options	17	deploying MobiLink applications	337
#hook_dict table		auth_status synchronization parameter	
dbmsync	269	about	100
A		authenticate_parameters	
ActiveSync		connection event	98
class name for dbmsync	81	authenticate_user	
installing the MobiLink provider	300	connection event	100
MobiLink ActiveSync provider		authenticate_user_hashed	
[dbasinst]	300	connection event	104
MobiLink clients using	252	authenticating	
ActiveSync provider installation utility		users in MobiLink	308
[dbasinst]		auto-dial	
syntax	300	MobiLink clients using HTTP	248
Adaptive Server Anywhere clients		MobiLink clients using HTTPS	250
		MobiLink clients using TCP/IP	246

B

backups	
restoring remote databases	77
begin_connection	
connection event	107
begin_connection_autocommit	
connection event	109
begin_download	
connection event	110
table event	112
begin_download_deletes	
table event	114
begin_download_rows	
table event	116
begin_publication	
connection event	118
begin_synchronization	
connection event	121
table event	123
begin_upload	
connection event	125
table event	127
begin_upload_deletes	
table event	129
begin_upload_rows	
table event	131
buffer_size stream parameter	
MobiLink clients using HTTP	247
MobiLink clients using HTTPS	250

C

C#	
MobiLink options	17
cache size	
dbmsync upload stream	56
certificate generation utility [gencert]	
syntax	311
certificate reader utility [readcert]	
syntax	310
certificate stream parameter	
HTTPS synchronization	27
certificate_company stream parameter	
MobiLink clients using HTTP	249
MobiLink clients using HTTPS	251
MobiLink clients using TCP/IP	246
certificate_name stream parameter	

MobiLink clients using HTTP	249
MobiLink clients using HTTPS	251
MobiLink clients using TCP/IP	246
certificate_password stream parameter	
HTTPS synchronization	28
certificate_unit stream parameter	
MobiLink clients using HTTP	249
MobiLink clients using HTTPS	251
MobiLink clients using TCP/IP	246
certificates	
generating elliptic-curve	311
generating RSA	311
reading elliptic-curve	310
reading RSA	310
character sets	
MobiLink synchronization	332
character-set translation	
by ODBC drivers	333
during MobiLink synchronization	
under other platforms	334
during MobiLink synchronization	
under Windows	332
class names	
ActiveSync	81
client database extraction utility	
[mlxtract]	
syntax	304
client event-hook procedures	269
client_port number	
default for MobiLink clients using	
HTTPS	250
client_port stream parameter	
HTTP synchronization	25
MobiLink clients using HTTP	247
MobiLink clients using TCP/IP	245
clients	
dbmsync	36
CLR	
MobiLink options	17
collation sequences	
MobiLink synchronization	332
command line	
starting dbmsrv9	4
starting dbmsync	36
command line utilities	
dbasinst command line syntax	300
MobiLink certificate generator	

-
- | | | | |
|---------------------------------------|-----|---------------------------------------|-----|
| [gencert] | 311 | MobiLink synchronization | 330 |
| MobiLink client database extraction | | to Oracle data types in MobiLink | |
| [mlxtract] | 304 | synchronization | 328 |
| MobiLink stop utility [dbmlstop] | 303 | cr | |
| MobiLink synchronization | 299 | dbmlsync extended option | 47 |
| MobiLink user authentication | | CREATE PUBLICATION statement | |
| [dbmluser] | 308 | SQL syntax | 240 |
| readcert syntax | 310 | CREATE SYNCHRONIZATION | |
| COMMIT statement | | SUBSCRIPTION statement | |
| event-hook procedures | 269 | SQL syntax | 243 |
| common language runtime | | CREATE SYNCHRONIZATION USER | |
| MobiLink options | 17 | statement | |
| communication protocols | | SQL syntax | 245 |
| multiple settings in MobiLink | 252 | creating | |
| CommunicationAddress | | MobiLink client databases | 304 |
| dbmlsync extended option | 45 | new certificates | 311 |
| CommunicationType | | publications | 240 |
| dbmlsync extended option | 46 | ctp | |
| complete event model | | dbmlsync extended option | 46 |
| MobiLink | 86 | | |
| ConflictRetries | | D | |
| dbmlsync extended option | 47 | data types | |
| connection scripts | | conversion of in MobiLink | |
| adding .NET scripts | 264 | synchronization | 323 |
| adding Java scripts | 266 | conversion to Adaptive Server | |
| adding SQL scripts | 262 | Enterprise in MobiLink | |
| deleting .NET scripts | 264 | synchronization | 324 |
| deleting Java scripts | 266 | conversion to IBM DB2 in MobiLink | |
| deleting SQL scripts | 262 | synchronization | 326 |
| consolidated databases | | conversion to Microsoft SQL Server in | |
| conversion of data types in MobiLink | | MobiLink synchronization | 330 |
| 323 | | conversion to Oracle in MobiLink | |
| MobiLink system tables | 316 | synchronization | 328 |
| contd_timeout stream parameter | | database extraction utility | |
| HTTP synchronization | 25 | MobiLink | 304 |
| HTTPS synchronization | 27 | DB2 | |
| conventions | | conversion of data types in MobiLink | |
| documentation | x | synchronization | 326 |
| conversion | | maximum identifier length in IBM | 316 |
| of data types in MobiLink | | dbasdesk.dll | |
| synchronization | 323 | installing | 300 |
| to Adaptive Server Enterprise data | | dbasdev.dll | |
| types in MobiLink | | installing | 300 |
| synchronization | 324 | dbasinst utility | |
| to IBM DB2 data types in MobiLink | | options | 300 |
| synchronization | 326 | syntax | 300 |
| to Microsoft SQL Server data types in | | dbmlsrv9 | |

options	4	sp_hook_dbmsync_schema_upgrade	
reports error context in output log	13	291	
dbmlstop utility		sp_hook_dbmsync_upload_begin	292
options	303	sp_hook_dbmsync_upload_end	293
syntax	303	sp_hook_dbmsync_validate_-	
dbmsync extended options	44	download_file	
dbmsync utility		295	
#hook_dict table	269	syntax	36
about	35	dbmluser utility	
event hooks	269	options	308
extended options	44	syntax	308
options	36	dbs	
sp_hook_dbmsync_abort hook	270	dbmsync extended option	48
sp_hook_dbmsync_begin	272	deleting	
sp_hook_dbmsync_delay	273	MobiLink .NET connection scripts	264
sp_hook_dbmsync_download_begin		MobiLink .NET table scripts	265
275		MobiLink Java connection scripts	266
sp_hook_dbmsync_download_com_-		MobiLink Java table scripts	267
error		MobiLink SQL connection scripts	262
275		MobiLink SQL table scripts	263
sp_hook_dbmsync_download_end		deploying	
276		Adaptive Server Anywhere MobiLink	
sp_hook_dbmsync_download_fatal_-		clients	342
sql_error		applications and databases	337
277		MobiLink applications	337
sp_hook_dbmsync_download_log_-		MobiLink synchronization server	339
ri_violation		UltraLite applications	344
279		dir	
sp_hook_dbmsync_download_ri_-		dbmsync extended option	58
violation		DisablePolling	
280		dbmsync extended option	47
sp_hook_dbmsync_download_sql_-		documentation	
error		conventions	x
282		SQL Anywhere Studio	viii
sp_hook_dbmsync_download_table_-		download events	
begin		MobiLink synchronization	96
283		download only synchronization	
sp_hook_dbmsync_download_table_-		dbmsync -ds option	43
end		download stream	
283		-uo option for upload-only	
sp_hook_dbmsync_end	284	synchronization	80
sp_hook_dbmsync_log_rescan	286	download-only synchronization	
sp_hook_dbmsync_logscan_begin	287	Adaptive Server Anywhere remote	
sp_hook_dbmsync_logscan_end	288	databases	49
sp_hook_dbmsync_process_return_-		download_cursor	
code		cursor event	133
289		download_delete_cursor	

cursor event	136	end_upload	
download_statistics		connection event	162
connection event	139	table event	164
table event	142	end_upload_deletes	
DownloadBufferSize		table event	166
dbmsync extended option	48	end_upload_rows	
downloading rows		table event	168
resolving MobiLink RI violations	279	error logs	
DownloadOnly		MobiLink server [dbmlsrv9]	12
dbmsync extended option	49	ErrorLogSendLimit	
drivers		dbmsync extended option	50
MobiLink ODBC drivers	336	event hooks	
DROP PUBLICATION statement		commits not allowed	269
SQL syntax	255	rollbacks not allowed	269
DROP SYNCHRONIZATION		sp_hook_dbmsync_abort	270
SUBSCRIPTION statement		sp_hook_dbmsync_begin	272
SQL syntax	256	sp_hook_dbmsync_delay	273
DROP SYNCHRONIZATION USER		sp_hook_dbmsync_download_begin	
statement		275	
SQL syntax	257	sp_hook_dbmsync_download_com_-	
dropping		error	
publications	255	275	
ds		sp_hook_dbmsync_download_fatal_-	
dbmsync extended option	49	SQL_error	
E		277	
eh		sp_hook_dbmsync_download_log_-	
dbmsync extended option	53	ri_violation	
el		279	
dbmsync extended option	50	sp_hook_dbmsync_download_ri_-	
elliptic-curve certificates		violation	
generating	311	280	
reading	310	sp_hook_dbmsync_download_sql_-	
end_connection		error	
connection event	145	282	
end_download		sp_hook_dbmsync_download_table_-	
connection event	147	begin	
table event	149	283	
end_download_deletes		sp_hook_dbmsync_download_table_-	
table event	151	end	
end_download_rows		283	
table event	153	sp_hook_dbmsync_end	284
end_publication		sp_hook_dbmsync_log_rescan	286
connection event	155	sp_hook_dbmsync_logscan_begin	287
end_synchronization		sp_hook_dbmsync_logscan_end	288
connection event	158	sp_hook_dbmsync_process_return_-	
table event	160	code	
		289	

sp_hook_dbmlsync_upload_begin	291,	elliptic-curve certificates	311
292		RSA certificates	311
sp_hook_dbmlsync_upload_end	293		
sp_hook_dbmlsync_validate_-		H	
download_file		handle_error	
295		connection event	174
synchronization event hooks	269	handle_odbc_error	
event-hooks		connection event	177
sp_hook_dbmlsync_begin	275	hooks	
sp_hook_dbmlsync_download_end		ignoring errors	53
276		sp_hook_dbmlsync_abort	270
events		sp_hook_dbmlsync_begin	272
about MobiLink synchronization	86	sp_hook_dbmlsync_delay	273
MobiLink	83	sp_hook_dbmlsync_download_begin	
example_upload_cursor		275	
cursor event	170	sp_hook_dbmlsync_download_com_-	
example_upload_delete		error	
table event	171	275	
example_upload_insert		sp_hook_dbmlsync_download_end	
table event	172	276, 284	
example_upload_update		sp_hook_dbmlsync_download_fatal_-	
table event	173	sql_error	
extended options		277	
dbmlsync	44	sp_hook_dbmlsync_download_log_-	
extracting		ri_violation	
MobiLink client databases	304	279	
extraction utility		sp_hook_dbmlsync_download_ri_-	
MobiLink	304	violation	
		280	
F		sp_hook_dbmlsync_download_sql_-	
feedback		error	
documentation	xiv	282	
providing	xiv	sp_hook_dbmlsync_download_table_-	
file-based downloads		begin	
dbmlsync -bc option	40	283	
dbmlsync -be option	41	sp_hook_dbmlsync_download_table_-	
dbmlsync -bg option	41	end	
FireTriggers		283	
dbmlsync extended option	51	sp_hook_dbmlsync_log_rescan	286
ft		sp_hook_dbmlsync_logscan_begin	287
dbmlsync extended option	51	sp_hook_dbmlsync_logscan_end	288
		sp_hook_dbmlsync_process_return_-	
G		code	
gencert utility		289	
options	311	sp_hook_dbmlsync_schema_upgrade	
syntax	311	291	
generating		sp_hook_dbmlsync_upload_begin	292

ml_add_java_table_script stored procedure		MobiLink clients	
SQL syntax	267	deploying	342
ml_add_table_script stored procedure		MobiLink events	
SQL syntax	263	listed	83
ml_connection_script		MobiLink performance	
MobiLink system table	317	estimate number of upload rows	80
ml_script		MobiLink security	
MobiLink system table	317	custom user authentication	104
ml_script_version		MobiLink stop utility [dbmlstop]	
MobiLink system table	318	syntax	303
ml_scripts_modified		MobiLink synchronization	
MobiLink system table	318	overview of events	86
ml_subscription		schedule option syntax	59
MobiLink system table	318	MobiLink synchronization client	
ml_table		dbmlsync options	36
MobiLink system table	319	MobiLink synchronization server	
ml_table_script		deploying	339
MobiLink system table	320	options	4
ml_user		stopping	303
MobiLink system table	320	switches	4
mlxtract utility		syntax	4
options	304	MobiLink system tables	
syntax	304	about	316
mn		MobiLink user authentication utility	
dbmlsync extended option	57	[dbmluser]	
MobiLink		syntax	308
complete event model	86	MobiLink utilities	
creating publications	240	MobiLink ActiveSync provider	
events	83	[dbasinst]	300
logging RI violations	279	MobiLink certificate generator	
ODBC drivers	336	[gencert]	311
overview of event process	86	MobiLink certificate reader [readcert]	
schedule option syntax	59	310	
stopping	303	MobiLink client database extraction	
MobiLink ActiveSync provider		[mlxtract]	304
installation utility [dbasinst]		MobiLink stop utility [dbmlstop]	303
syntax	300	MobiLink user authentication	
MobiLink certificate generation utility		[dbmluser]	308
[gencert]		MobiLinkPwd	
syntax	311	dbmlsync extended option	56
MobiLink certificate reader utility		modify_last_download_timestamp	
[readcert]		connection event	180
syntax	310	modify_next_last_download_timestamp	
MobiLink client database extraction		connection event	182
utility [mlxtract]		modify_user	
syntax	304	connection event	184
		monitoring	

-
- logging MobiLink RI violations 279
 - mp
 - dbmlsync extended option 56
 - N**
 - network parameters
 - on the dbmlsync command line 45
 - network_connect_timeout stream parameter
 - MobiLink clients using HTTP 248
 - MobiLink clients using HTTPS 250
 - MobiLink clients using TCP/IP 246
 - network_leave_open stream parameter
 - MobiLink clients using HTTP 248
 - MobiLink clients using HTTPS 250
 - MobiLink clients using TCP/IP 246
 - network_name stream parameter
 - MobiLink clients using HTTP 248
 - MobiLink clients using HTTPS 250
 - MobiLink clients using TCP/IP 246
 - new_row_cursor
 - cursor event 186
 - NewMobiLinkPwd
 - dbmlsync extended option 57
 - newsgroups
 - technical support xiv
 - O**
 - ODBC drivers
 - for use with MobiLink 336
 - MobiLink character-set translation by 333
 - OfflineDirectory
 - dbmlsync extended option 58
 - offset
 - about 77
 - old_row_cursor
 - cursor event 189
 - OPTION clause
 - CREATE SYNCHRONIZATION USER 252
 - options
 - dbmlsrv9 4
 - dbmlsync 36
 - dbmlsync extended options 44
 - extended options for dbmlsync 44
 - MobiLink ActiveSync provider
 - [dbasinst] 300
 - MobiLink certificate generator
 - [gencert] 311
 - MobiLink certificate reader [readcert] 310
 - MobiLink client [dbmlsync] 36
 - MobiLink client database extraction
 - [mlxtract] 304
 - MobiLink server [dbmlsrv9] 4
 - MobiLink stop utility [dbmlstop] 303
 - MobiLink synchronization clients 252
 - MobiLink user authentication
 - [dbmluser] 308
 - Oracle
 - conversion of data types in MobiLink synchronization 328
 - data types 328
 - ODBC configuration 329
 - synchronizing LONG data 329
 - P**
 - p
 - dbmlsync extended option 47
 - passwords
 - MobiLink dbmluser utility 308
 - persistent stream parameter
 - MobiLink clients using HTTP 248
 - MobiLink clients using HTTPS 251
 - pinging
 - MobiLink synchronization server 76
 - polling
 - dbmlsync logscan polling 47
 - PollingPeriod
 - dbmlsync extended option 58
 - port stream parameter
 - HTTP synchronization 26
 - HTTPS synchronization 27
 - MobiLink clients using HTTP 248
 - MobiLink clients using HTTPS 251
 - MobiLink clients using TCP/IP 246
 - TCP/IP synchronization 25
 - PP
 - dbmlsync extended option 58
 - prepare_for_download
 - connection event 192
 - progress

about	77	reading	310
protocols		S	
HTTP synchronization	25	sa	
HTTPS synchronization	27	dbmlsync extended option	62
MobiLink clients using ActiveSync	252	sch	
MobiLink clients using HTTP	247	dbmlsync extended option	59
MobiLink clients using HTTPS	249	Schedule	
MobiLink clients using TCP/IP	245	dbmlsync extended option	59
TCP/IP synchronization	24	scheduling	
proxy_host stream parameter		ignore for dbmlsync	53
MobiLink clients using HTTP	248	MobiLink schedule option syntax	59
MobiLink clients using HTTPS	251	scripts	
proxy_port stream parameter		adding and deleting .NET connection	
MobiLink clients using HTTP	248	scripts	264
MobiLink clients using HTTPS	251	adding and deleting .NET table scripts	
pseudocode		265	
MobiLink events	86	adding and deleting Java connection	
publications		scripts	266
altering	234	adding and deleting Java table scripts	
creating	240	267	
dropping	255	adding and deleting SQL connection	
offsets	77	scripts	262
R		adding and deleting SQL table scripts	
readcert utility		263	
options	310	MobiLink events	83
syntax	310	ScriptVersion	
reading		dbmlsync extended option	61
elliptic-curve certificates	310	security	
RSA certificates	310	MobiLink custom user authentication	
referential integrity		100	
resolving MobiLink RI violations	279	security stream parameter	
remote databases		MobiLink clients using HTTP	249
restoring from backup	77	MobiLink clients using TCP/IP	246
report_error		SendColumnNames	
connection event	194	dbmlsync extended option	62
report_odbc_error		SendDownloadACK	
connection event	196	dbmlsync extended option	62
resolve_conflict		SendTriggers	
table event	199	dbmlsync extended option	63
restoring		server stored procedures	
remote databases from backup	77	MobiLink	262
ROLLBACK statement		signing	
event-hook procedures	269	elliptic-curve and RSA certificates	311
RSA certificates		sort order	
generating	311	characters and MobiLink	
		synchronization	332

sp_hook_dbmlsync_abort stored procedure		procedure	
SQL syntax	270	SQL syntax	287
sp_hook_dbmlsync_begin stored procedure		sp_hook_dbmlsync_logscan_end stored procedure	
SQL syntax	272	SQL syntax	288
sp_hook_dbmlsync_delay stored procedure		sp_hook_dbmlsync_process_return_code stored procedure	
SQL syntax	273	SQL syntax	289
sp_hook_dbmlsync_download_begin stored procedure		sp_hook_dbmlsync_schema_upgrade stored procedure	
SQL syntax	275	SQL syntax	291
sp_hook_dbmlsync_download_com_-error stored procedure		sp_hook_dbmlsync_upload_begin stored procedure	
SQL syntax	275	SQL syntax	292
sp_hook_dbmlsync_download_end stored procedure		sp_hook_dbmlsync_upload_end stored procedure	
SQL syntax	276	SQL syntax	293
sp_hook_dbmlsync_download_fatal_-SQL_error stored procedure		sp_hook_dbmlsync_validate_download_-file stored procedure	
SQL syntax	277	SQL syntax	295
sp_hook_dbmlsync_download_log_ri_-violation stored procedure		SQL Anywhere Studio documentation	viii
SQL syntax	279	SQL Remote creating publications	240
sp_hook_dbmlsync_download_ri_-violation stored procedure		SQL statements	
SQL syntax	280	ALTER PUBLICATION syntax	234
sp_hook_dbmlsync_download_sql_error stored procedure		ALTER SYNCHRONIZATION SUBSCRIPTION syntax	236
SQL syntax	282	ALTER SYNCHRONIZATION USER syntax	238
sp_hook_dbmlsync_download_table_-begin stored procedure		CREATE PUBLICATION syntax	240
SQL syntax	283	CREATE SYNCHRONIZATION SUBSCRIPTION syntax	243
sp_hook_dbmlsync_download_table_end stored procedure		CREATE SYNCHRONIZATION USER syntax	245
SQL syntax	283	DROP PUBLICATION syntax	255
sp_hook_dbmlsync_end stored procedure		DROP SYNCHRONIZATION SUBSCRIPTION syntax	256
SQL syntax	284	DROP SYNCHRONIZATION USER syntax	257
sp_hook_dbmlsync_log_rescan stored procedure		START SYNCHRONIZATION DELETE syntax	258
SQL syntax	286	STOP SYNCHRONIZATION DELETE syntax	260
sp_hook_dbmlsync_logscan_begin stored procedure		SQL syntax	
		ml_add_connection_script stored	

procedure	262	sp_hook_dbmlsync_logscan_begin	
ml_add_dnet_connection_script stored		stored procedure	287
procedure	264	sp_hook_dbmlsync_logscan_end	
ml_add_dnet_table_script stored		stored procedure	288
procedure	265	sp_hook_dbmlsync_process_return_-	
ml_add_java_connection_script stored		code stored procedure	289
procedure	266	sp_hook_dbmlsync_upload_begin	
ml_add_java_table_script stored		stored procedure	292
procedure	267	sp_hook_dbmlsync_upload_end	
ml_add_table_script stored procedure		stored procedure	293
263		sp_hook_dbmlsync_upload_schema_-	
MobiLink server [dbmlsrv9]	4	upgrade procedure	
sp_hook_dbmlsync_abort stored		291	
procedure	270	sp_hook_dbmlsync_validate_-	
sp_hook_dbmlsync_begin stored		download_file stored procedure	
procedure	272	295	
sp_hook_dbmlsync_delay stored		st	
procedure	273	dbmlsync extended option	63
sp_hook_dbmlsync_download_begin		start classes	
stored procedure	275	DMLStartClasses option for Java	19
sp_hook_dbmlsync_download_com_-		MLStartClasses option for .NET	17
error stored procedure		START SYNCHRONIZATION DELETE	
275		statement	
sp_hook_dbmlsync_download_end		SQL syntax	258
stored procedure	276	statements	
sp_hook_dbmlsync_download_fatal_-		ALTER PUBLICATION syntax	234
SQL_error stored procedure		ALTER SYNCHRONIZATION	
277		SUBSCRIPTION syntax	236
sp_hook_dbmlsync_download_log_-		ALTER SYNCHRONIZATION USER	
ri_violation stored procedure		syntax	238
279		CREATE PUBLICATION syntax	240
sp_hook_dbmlsync_download_ri_-		CREATE SYNCHRONIZATION	
violation stored procedure		SUBSCRIPTION syntax	243
280		CREATE SYNCHRONIZATION	
sp_hook_dbmlsync_download_sql_-		USER syntax	245
error stored procedure		DROP PUBLICATION syntax	255
282		DROP SYNCHRONIZATION	
sp_hook_dbmlsync_download_table_-		SUBSCRIPTION syntax	256
begin stored procedure		DROP SYNCHRONIZATION USER	
283		syntax	257
sp_hook_dbmlsync_download_table_-		START SYNCHRONIZATION	
end stored procedure		DELETE syntax	258
283		STOP SYNCHRONIZATION	
sp_hook_dbmlsync_end stored		DELETE syntax	260
procedure	284	STOP SYNCHRONIZATION DELETE	
sp_hook_dbmlsync_log_rescan stored		statement	
procedure	286		

-
- SQL syntax 260
 - stop utility [dbmlstop]
 - syntax 303
 - stopping
 - MobiLink 303
 - stored procedures
 - ml_add_connection_script SQL syntax 262
 - ml_add_dnet_connection_script SQL syntax 264
 - ml_add_dnet_table_script SQL syntax 265
 - ml_add_java_connection_script SQL syntax 266
 - ml_add_java_table_script SQL syntax 267
 - ml_add_table_script SQL syntax 263
 - MobiLink 261
 - MobiLink client procedures 269
 - MobiLink server 262
 - sp_hook_dbmlsync_abort SQL syntax 270
 - sp_hook_dbmlsync_begin SQL syntax 272
 - sp_hook_dbmlsync_delay SQL syntax 273
 - sp_hook_dbmlsync_download_begin SQL syntax 275
 - sp_hook_dbmlsync_download_com_-error SQL syntax 275
 - sp_hook_dbmlsync_download_end SQL syntax 276
 - sp_hook_dbmlsync_download_fatal_-SQL_error SQL syntax 277
 - sp_hook_dbmlsync_download_log_-ri_violation 279
 - sp_hook_dbmlsync_download_ri_-violation 280
 - sp_hook_dbmlsync_download_sql_-error SQL syntax 282
 - sp_hook_dbmlsync_download_table_-begin SQL syntax 283
 - sp_hook_dbmlsync_download_table_-end SQL syntax 283
 - sp_hook_dbmlsync_end SQL syntax 284
 - sp_hook_dbmlsync_log_rescan SQL syntax 286
 - sp_hook_dbmlsync_logscan_begin SQL syntax 287
 - sp_hook_dbmlsync_logscan_end SQL syntax 288
 - sp_hook_dbmlsync_process_return_-code SQL syntax 289
 - sp_hook_dbmlsync_schema_upgrade SQL syntax 291
 - sp_hook_dbmlsync_upload_begin SQL syntax 292
 - sp_hook_dbmlsync_upload_end SQL syntax 293
 - sp_hook_dbmlsync_validate_-download_file SQL syntax 295
 - SUBSCRIBE BY clause 240
 - support
 - newsgroups xiv
 - sv
 - dbmlsync extended option 61, 62
 - switches
 - MobiLink ActiveSync provider [dbasinst] 300
 - MobiLink certificate generator [gencert] 311
 - MobiLink certificate reader [readcert] 310
 - MobiLink client [dbmlsync] 36
 - MobiLink client database extraction [mlxtract] 304
 - MobiLink server [dbmlsrv9] 4
 - MobiLink user authentication [dbmluser] 308
 - Sybase Adaptive Server Enterprise
 - conversion of data types in MobiLink synchronization 324
 - synchronization
 - conversion of data types in MobiLink

323		end_publication	155
conversion to Adaptive Server		end_synchronization	158, 160
Enterprise data types in		end_upload	162, 164
MobiLink	324	end_upload_deletes	166
conversion to IBM DB2 data types in		end_upload_rows	168
MobiLink	326	example_upload_cursor	170
conversion to Microsoft SQL Server		example_upload_delete	171
data types in MobiLink	330	example_upload_insert	172
conversion to Oracle data types in		example_upload_update	173
MobiLink	328	handle_error	174
customizing	269	handle_odbc_error	177
event hooks	269	MobiLink download	96
MobiLink character sets	332	MobiLink upload	92
MobiLink character-set translation		modify_last_download_timestamp	180
under other platforms	334	modify_next_last_download_	
MobiLink character-set translation		timestamp	
under Windows	332	182	
MobiLink stored procedures	262	modify_user	184
MobiLink system tables	316	new_row_cursor	186
MobiLink utilities	299	old_row_cursor	189
schedule option syntax for MobiLink		prepare_for_download	192
59		report_error	194
transactions	269	report_odbc_error	196
synchronization errors		resolve_conflict	199
troubleshooting	12	synchronization_statistics	202, 205
synchronization events		time_statistics	207, 209
about MobiLink synchronization	86	upload_cursor	212
authenticate_parameters	98	upload_delete	214
authenticate_user	100	upload_fetch	216
authenticate_user_hashed	104	upload_insert	218
begin_connection	107	upload_new_row_insert	220
begin_connection_autocommit	109	upload_old_row_insert	222
begin_download	110, 112	upload_statistics	224, 227
begin_download_deletes	114	upload_update	231
begin_download_rows	116	synchronization parameters	
begin_publication	118	HTTP synchronization	24
begin_synchronization	121, 123	HTTPS synchronization	24
begin_upload	125, 127	TCP/IP synchronization	24
begin_upload_deletes	129	synchronization scripts	
begin_upload_rows	131	MobiLink events	83
download_cursor	133	synchronization_statistics	
download_delete_cursor	136	connection event	202
download_statistics	139, 142	table event	205
end_connection	145	syntax	
end_download	147, 149	MobiLink ActiveSync provider	
end_download_deletes	151	[dbasinst]	300
end_download_rows	153	MobiLink certificate generator	

- [gencert] 311
- MobiLink certificate reader [readcert] 310
- MobiLink client database extraction [mlxtract] 304
- MobiLink scripts 83
- MobiLink stop utility [dbmlstop] 303
- MobiLink stored procedures 262
- MobiLink synchronization utilities 299
- MobiLink user authentication [dbmluser] 308
- sys.servers system table
 - remote servers for Component Integration Services 253
- system tables
 - MobiLink synchronization 316
- T**
- table scripts
 - adding .NET scripts 265
 - adding Java scripts 267
 - adding SQL scripts 263
 - deleting .NET scripts 265
 - deleting Java scripts 267
 - deleting SQL scripts 263
- TableOrder
 - dbmlsync extended option 64
- TCP/IP
 - dbmlsrv9 -x command line option 24
 - MobiLink clients using synchronization parameters 245
- technical support
 - newsgroups xiv
- time_statistics
 - connection event 207
 - table event 209
- tor
 - dbmlsync extended option 64
- translation
 - character-set by ODBC drivers 333
- translation between character sets
 - MobiLink synchronization under other platforms 334
 - MobiLink synchronization under Windows 332
- troubleshooting
- restoring the remote database from
 - backup 77
 - synchronization errors 12
- trusted_certificates stream parameter
 - MobiLink clients using HTTP 249
 - MobiLink clients using HTTPS 251
 - MobiLink clients using TCP/IP 246
- TYPE clause
 - CREATE SYNCHRONIZATION USER 245
- U**
- UltraLite
 - deploying 344
- unknown_timeout stream parameter
 - HTTP synchronization 26
 - HTTPS synchronization 28
- uo
 - dbmlsync extended option 65
- upload events
 - MobiLink synchronization 92
- upload only synchronization
 - dbmlsync -uo option 80
- upload stream
 - uo option for upload-only synchronization 80
- upload-only synchronization
 - Adaptive Server Anywhere remote databases 65
- upload_cursor
 - cursor event 212
- upload_delete
 - table event 214
- upload_fetch
 - table event 216
- upload_insert
 - table event 218
- upload_new_row_insert
 - table event 220
- upload_old_row_insert
 - table event 222
- upload_statistics
 - connection event 224
 - table event 227
- upload_update
 - table event 231
- UploadOnly

dbmlsync extended option	65	MobiLink [dbmlsync]	80
url_suffix stream parameter		version stream parameter	
HTTP synchronization	26	HTTP synchronization	27
HTTPS synchronization	28	HTTPS synchronization	28
MobiLink clients using HTTP	249	MobiLink clients using HTTP	249
MobiLink clients using HTTPS	251	MobiLink clients using HTTPS	252
user authentication utility [dbmluser]		vm	
syntax	308	dbmlsync extended option	67
user names		vn	
MobiLink user authentication utility		dbmlsync extended option	68
[dbmluser]	308	vo	
utilities		dbmlsync extended option	68
MobiLink ActiveSync provider		vr	
[dbasinst]	300	dbmlsync extended option	69
MobiLink certificate generator		vs	
[gencert]	311	dbmlsync extended option	66
MobiLink certificate reader [readcert]		vu	
310		dbmlsync extended option	70
MobiLink client database extraction			
[mlxtract]	304	W	
MobiLink stop utility [dbmlstop]	303	Windows CE	
MobiLink synchronization	299	dbmlsync applications	75
MobiLink user authentication		X	
[dbmluser]	308	X509 certificates	
V		generating	311
v		reading	310
dbmlsync extended option	65	Xusage.txt	
Verbose		location	19
dbmlsync extended option	65		
VerboseHooks			
dbmlsync extended option	66		
VerboseMin			
dbmlsync extended option	67		
VerboseOptions			
dbmlsync extended option	68		
VerboseRowCounts			
dbmlsync extended option	68		
VerboseRowValues			
dbmlsync extended option	69		
VerboseUpload			
dbmlsync extended option	70		
verbosity			
setting in MobiLink [dbmlsrv9]	21		
setting in MobiLink [dbmlsync]	80		
verbosity option			
MobiLink [dbmlsrv9]	21		