



# MobiLink Tutorials

Part number: DC00194-01-0902-01  
Last modified: October 2004

---

Copyright © 1989–2004 Sybase, Inc. Portions copyright © 2001–2004 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, 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, E-Whatever, 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, Intelligent Self-Care, InternetBuilder, iremote, iScript, Jaguar CTS, jConnect for JDBC, KnowledgeBase, Logical Memory Manager, Mail Anywhere Studio, MainframeConnect, Maintenance Express, Manage Anywhere Studio, MAP, M-Business Channel, M-Business Network, M-Business Server, 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++, Orchestration Studio, Partnerships that Work, PB-Gen, PC APT Execute, PC DB-Net, PC Net Library, PhysicalArchitect, Pocket PowerBuilder, PocketBuilder, Power Through Knowledge, power.stop, Power++, 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, SKILLS, 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, TotalFix, 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


<b>About This Manual</b>	<b>v</b>
SQL Anywhere Studio documentation . . . . .	vi
Documentation conventions . . . . .	ix
Finding out more and providing feedback . . . . .	xi
<b>1 Tutorial: Introduction to MobiLink</b>	<b>1</b>
Introduction . . . . .	2
Lesson 1: Creating and populating your databases . . . . .	3
Lesson 2: Running the MobiLink synchronization server . . . . .	6
Lesson 3: Running the MobiLink synchronization client . . . . .	8
Cleanup . . . . .	10
Summary . . . . .	11
Further reading . . . . .	12
<b>2 Tutorial: Writing MobiLink Scripts and Monitoring Synchronizations</b>	<b>13</b>
Introduction . . . . .	14
Lesson 1: Set up the Adaptive Server Anywhere consolidated database . . . . .	15
Lesson 2: Set up the remote Adaptive Server Anywhere databases . . . . .	18
Lesson 3: Creating scripts for your synchronization . . . . .	22
Lesson 4: Run MobiLink synchronization . . . . .	25
Lesson 5: Monitoring your MobiLink synchronization using log files . . . . .	26
Lesson 6: Creating scripts for conflict detection and resolution . . . . .	28
Lesson 7: Use the MobiLink Monitor to detect update conflicts . . . . .	31
Tutorial cleanup . . . . .	37
Further reading . . . . .	38
<b>3 Tutorial: Using MobiLink with an Oracle 8i Consolidated Database</b>	<b>39</b>
Introduction . . . . .	40
Lesson 1: Create your databases . . . . .	41
Lesson 2: Running the MobiLink synchronization server . . . . .	47
Lesson 3: Running the MobiLink synchronization client . . . . .	48
Summary . . . . .	49
Further reading . . . . .	50

---

<b>4</b>	<b>Tutorial: Java Synchronization Logic With Adaptive Server Anywhere</b>	<b>51</b>
	Introduction . . . . .	52
	Lesson 1: Compiling the CustdbScripts Java class . . . . .	53
	Lesson 2: Specifying class methods for events . . . . .	55
	Lesson 3: Run the MobiLink server with -sl java . . . . .	60
	Lesson 4: Test synchronization . . . . .	61
	Cleanup . . . . .	63
	Further reading . . . . .	64
<b>5</b>	<b>Tutorial: .NET Synchronization Logic With Adaptive Server Anywhere</b>	<b>65</b>
	Introduction . . . . .	66
	Lesson 1: Compile the CustdbScripts.dll assembly with Mo- biLink references . . . . .	67
	Lesson 2: Specify class methods for events . . . . .	72
	Lesson 3: Run MobiLink with -sl dnet . . . . .	77
	Lesson 4: Test synchronization . . . . .	78
	Cleanup . . . . .	80
	Further reading . . . . .	81
<b>6</b>	<b>The Contact Sample Application</b>	<b>83</b>
	Introduction . . . . .	84
	Setup . . . . .	85
	Tables in the Contact databases . . . . .	87
	Users in the Contact sample . . . . .	90
	Synchronization . . . . .	91
	Monitoring statistics and errors in the Contact sample . . . .	98
<b>7</b>	<b>The CustDB Sample Application</b>	<b>99</b>
	Introduction . . . . .	100
	Setup . . . . .	102
	Tables in the CustDB databases . . . . .	109
	Users in the CustDB sample . . . . .	112
	Synchronization . . . . .	113
	Maintaining the customer and order primary key pools . . . .	117
	Further reading . . . . .	119
	<b>Index</b>	<b>121</b>

---

# About This Manual

Subject	This manual contains tutorials that help you learn how to use MobiLink synchronization technology.
Before you begin	 For more information about MobiLink, see <a href="#">“Introducing MobiLink Synchronization”</a> [ <i>MobiLink Administration Guide</i> , page 3].

---

# 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 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 Administration 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 Clients** This book describes how to set up and synchronize Adaptive Server Anywhere and UltraLite remote databases.
- ◆ **MobiLink Server-Initiated Synchronization User's Guide** This book describes MobiLink server-initiated synchronization, a feature of MobiLink that allows you to initiate synchronization from the consolidated database.
- ◆ **QAnywhere User's Guide** This manual describes MobiLink QAnywhere, a messaging platform that enables the development and deployment of messaging applications for mobile and wireless clients, as well as traditional desktop and laptop clients.
- ◆ **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.

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

The PDF books are accessible from the online books, or from the Windows Start menu.

- ◆ **Printed books**    The complete set of books is available from Sybase sales or from eShop, the Sybase online store at <http://eshop.sybase.com/eshop/documentation>.



---

# 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 } ]
```

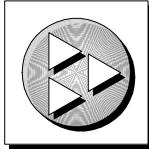
If the QUOTES option is used, one of ON or OFF must be provided. The brackets and braces should not be typed.

---

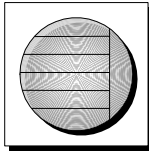
Graphic icons

The following icons are used in this documentation.

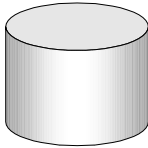
- ◆ A client application.



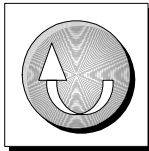
- ◆ A database server, such as Sybase Adaptive Server Anywhere.



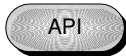
- ◆ A database. In some high-level diagrams, the icon may be used to represent both the database and the database server that manages it.



- ◆ 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.



---

## Finding out more and providing feedback

### Finding out more

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

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

When you write to one of these newsgroups, always provide detailed information about your problem, including the build number of your version of SQL Anywhere Studio. You can find this information by typing **dbeng9 -v** at a command prompt.

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

- ◆ [sybase.public.sqlanywhere.general](#)
- ◆ [sybase.public.sqlanywhere.linux](#)
- ◆ [sybase.public.sqlanywhere.mobilink](#)
- ◆ [sybase.public.sqlanywhere.product\\_futures\\_discussion](#)
- ◆ [sybase.public.sqlanywhere.replication](#)
- ◆ [ianywhere.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 ensure its operation and availability.

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

### Feedback

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

You can e-mail comments and suggestions to the SQL Anywhere documentation team at [iasdoc@ianywhere.com](mailto:iasdoc@ianywhere.com). Although we do not reply to e-mails sent to that address, we read all suggestions with interest.

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



---

## CHAPTER 1

# Tutorial: Introduction to MobiLink

About this chapter

This chapter provides a very introductory tutorial to guide you through the process of setting up a synchronization system that uses Adaptive Server Anywhere databases.

Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Introduction</a>	2
<a href="#">Lesson 1: Creating and populating your databases</a>	3
<a href="#">Lesson 2: Running the MobiLink synchronization server</a>	6
<a href="#">Lesson 3: Running the MobiLink synchronization client</a>	8
<a href="#">Cleanup</a>	10
<a href="#">Summary</a>	11
<a href="#">Further reading</a>	12

---

# Introduction

In this tutorial, you use Adaptive Server Anywhere to create a consolidated database and a remote database. You then synchronize these databases using MobiLink synchronization technology.

## Timing

The tutorial takes about 30 minutes.

## Goals

You will gain competence and familiarity with:

- ◆ The MobiLink synchronization server and client as an integrated system.
- ◆ The MobiLink synchronization server and client command lines and options.
- ◆ How to create an ODBC connection and set the properties of an ODBC connection for Adaptive Server Anywhere.
- ◆ How to initialize a database.

## Key concepts

The MobiLink synchronization server connects to the consolidated database using ODBC. The MobiLink synchronization client connects to your remote database. The MobiLink synchronization server and client function as a group, managing the upload and download of data from one database to another.

## Suggested background reading

For more information about the architecture of MobiLink, see [“Synchronization Basics”](#) [*MobiLink Administration Guide*, page 7].

☞ For more information about the Adaptive Server Anywhere utility for running SQL commands, see [“Using Interactive SQL”](#) [*Introducing SQL Anywhere Studio*, page 217].

# Lesson 1: Creating and populating your databases

MobiLink synchronization requires that you have a consolidated database, at least one remote database, and an ODBC data source for each database.

Create your database files

The first step is to create each of the databases. In this procedure, you build a consolidated database and a remote database using the `dbinit` utility from a command line.

The `dbinit` utility creates a database file with no user tables or procedures. You create your database schema when you define, within the newly-initialized file, user-defined tables and procedures.

## ❖ To create your database files

1. Open a command prompt and navigate to the `Samples\MobiLink\AutoScripting` subdirectory of your SQL Anywhere 9 installation.

2. Create a consolidated database for this tutorial. Run the following command line:

```
dbinit consol.db
```

If this tutorial has been previously run on your computer, `consol.db` and `consol.log` may already exist. This will cause `dbinit` to fail. Delete these files before running `dbinit`.

3. Create the remote database for this tutorial. Run the following command line:

```
dbinit remote.db
```

If this tutorial has been previously run on your computer, `remote.db` and `remote.log` may already exist. This will cause `dbinit` to fail. Delete these files before running `dbinit`.

4. Verify the successful creation of these database files by listing the contents of the directory. You should see `consol.db` and `remote.db` in the listing.

Create ODBC data sources

You are now ready to build ODBC data sources through which you can connect to your Adaptive Server Anywhere databases.

## ❖ To create ODBC data sources

1. Open a command prompt and navigate to the `Samples\MobiLink\AutoScripting` subdirectory of your SQL Anywhere 9 installation.

- 
2. Create an ODBC data source for your consolidated database by running the following command line:

```
dbdsn -w test_consol -y -c
      "uid=DBA;pwd=SQL;dbf=consol.db;eng=Consol"
```

This command line specifies the following dbdsn options:

- ◆ **-w** Creates a data source definition.
  - ◆ **-y** Delete or overwrite data source without confirmation.
  - ◆ **-c** Specifies the connection parameters as a connection string.
- ☞ For more information, see “Data Source utility options” [ASA *Database Administration Guide*, page 512].

3. Create an ODBC data source for a remote database by running the following command line:

```
dbdsn -w test_remote -y -c
      "uid=DBA;pwd=SQL;dbf=remote.db;eng=Remote"
```

## Create your schema

The following procedure executes SQL statements using the Interactive SQL utility to create and populate tables in the consolidated database. It also creates tables and inserts synchronization subscriptions and publications into the remote database.

It does this using two predefined SQL files, *build\_consol.sql* and *build\_remote.sql*. You may want to open these files in a text editor to examine them in detail.

### ❖ To create your schema

1. Open a command prompt and navigate to the *Samples\MobiLink\AutoScripting* subdirectory of your SQL Anywhere 9 installation.
2. Run the following command line:

```
dbisql -c "dsn=test_consol;astop=no" build_consol.sql
```

The SQL statements in *build\_consol.sql* create and populate the emp and cust tables in the consolidated database.

This step includes *astop=no* to instruct the server not to shut down when the *dbisql* utility shuts down.

3. Run the following command line:

```
dbisql -c "dsn=test_remote;astop=no" build_remote.sql
```

The SQL statements in *build\_remote.sql* create the remote tables emp and cust, and insert synchronization subscriptions and publications.



4. Verify the creation of the emp and cust tables in the remote and consolidated databases using Interactive SQL:
  - ◆ Open Interactive SQL by typing `dbisql` at a command prompt. Connect using the test\_consol DSN as DBA, using SQL as the password.
  - ◆ Execute the following SQL statement by typing it into the SQL Statements pane and pressing F9.

```
SELECT * FROM emp, cust
```

The tables in the consolidated database are populated with data.

- ◆ Connect using the test\_remote DSN and execute the following SQL statement:

```
SELECT * FROM emp, cust
```

The tables in the remote database are empty.

5. Leave the consolidated and remote databases running for the next lesson.

#### Further reading

- ☞ For more information about creating databases, see “[The Initialization utility](#)” [*ASA Database Administration Guide*, page 530] and “[Creating a database using the dbinit command-line utility](#)” [*ASA Database Administration Guide*, page 531].
- ☞ For more information about running Interactive SQL, see “[The Interactive SQL utility](#)” [*ASA Database Administration Guide*, page 538] and “[Using Interactive SQL](#)” [*Introducing SQL Anywhere Studio*, page 217].
- ☞ For more information about SELECT statements, see “[SELECT statement](#)” [*ASA SQL Reference*, page 597].
- ☞ For more information about creating ODBC data sources, see “[The Data Source utility](#)” [*ASA Database Administration Guide*, page 510].
- ☞ For more information about creating remote databases, see “[Creating a remote database](#)” [*MobiLink Clients*, page 60].
- ☞ For more information about creating subscriptions and publications, see “[Publishing data](#)” [*MobiLink Clients*, page 64].

---

## Lesson 2: Running the MobiLink synchronization server

Your consolidated database must be running prior to starting the MobiLink synchronization server. If you shut down your consolidated database following Lesson 1, you should restart the database.

### ❖ To start the MobiLink synchronization server

1. Open a command prompt and navigate to the `Samples\MobiLink\AutoScripting` subdirectory of your SQL Anywhere 9 installation.

2. Run the following command line:

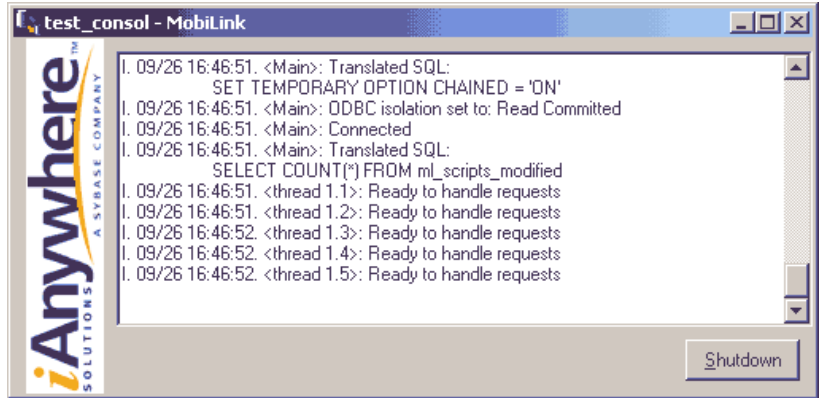
```
dbmlsrv9 -c "dsn=test_consol" -o mlserver.mls -v+ -dl -za -zu+
```

This command line specifies the following dbmlsrv9 options:

- ◆ **-c** The connection string for the MobiLink synchronization server uses the DSN for the consolidated database. For more information, see “[-c option](#)” [*MobiLink Administration Guide*, page 196].
- ◆ **-o** The `-o` option is used to specify the message log file. For more information, see “[-o option](#)” [*MobiLink Administration Guide*, page 203].
- ◆ **-v+** The `-v+` option sets verbose logging on. For more information, see “[-v option](#)” [*MobiLink Administration Guide*, page 211].
- ◆ **-dl** The `-dl` option sets the display log feature ON. For more information, see “[-dl option](#)” [*MobiLink Administration Guide*, page 199].
- ◆ **-za** The `-za` option turns automated scripting ON. For more information, see “[-za option](#)” [*MobiLink Administration Guide*, page 219].
- ◆ **-zu+** The `-zu+` option automates the user authentication process. For more information, see “[-zu option](#)” [*MobiLink Administration Guide*, page 222].

The options `-o`, `-v`, and `-dl` are chosen to provide debugging and troubleshooting information. These options are typically not used in a production environment.

Once you have executed the MobiLink synchronization server command, the output below appears.



If the MobiLink synchronization server is already running when you attempt to run `dbmlsrv9`, you will receive an error message. Shut down the current instance of MobiLink and run the command again.

#### Further reading

- ☞ For more information about the MobiLink synchronization server, see “[The MobiLink synchronization server](#)” [*MobiLink Administration Guide*, page 11].
- ☞ For a complete list of `dbmlsrv9` options, see “[MobiLink Synchronization Server Options](#)” [*MobiLink Administration Guide*, page 189].

---

## Lesson 3: Running the MobiLink synchronization client

Adaptive Server Anywhere clients initiate MobiLink synchronization by using the dbmlsync utility.

### ❖ To start the MobiLink synchronization client

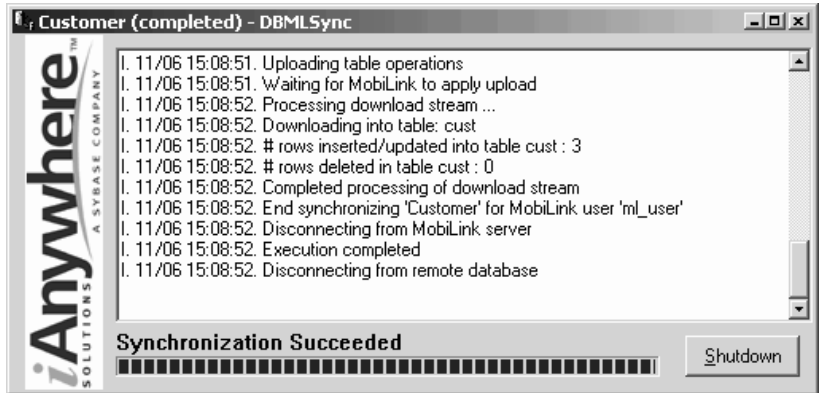
1. Open a command prompt and navigate to the *Samples\MobiLink\AutoScripting* subdirectory of your SQL Anywhere 9 installation.
2. Run the following command line:

```
dbmlsync -c "dsn=test_remote" -o dbmlsync.out -v -e  
"SendColumnNames=ON"
```

This command line specifies the following options:

- ◆ **-c** Supply database connection parameters. For more information, see “[-c option](#)” [*MobiLink Clients*, page 102].
- ◆ **-o** Specify the message log file. For more information, see “[-o option](#)” [*MobiLink Clients*, page 142].
- ◆ **-v** Verbose operation. For more information, see “[-v option](#)” [*MobiLink Clients*, page 150].
- ◆ **-e** Extended options. Specifying “SendColumnNames=ON” sends column names to MobiLink. This is required when you use `-za` in the `dbmlsrv9` command line. For more information, see “[SendColumnNames \(scn\) extended option](#)” [*MobiLink Clients*, page 130].

Once you have executed the MobiLink synchronization client command, the output below appears to indicate that synchronization has succeeded. After synchronization, the remote database is populated with the data from the consolidated database.



## Further reading

- ☞ For more information about dbmsync options, see [“MobiLink synchronization client”](#) [*MobiLink Clients*, page 96].
- ☞ For more information about remote clients, see [“MobiLink clients”](#) [*MobiLink Administration Guide*, page 14].
- ☞ For more information about dbmsync, see [“Initiating synchronization”](#) [*MobiLink Clients*, page 78].

---

## Cleanup

You should delete the databases and ODBC data sources that you created in this tutorial.

❖ **To remove tutorial materials from your computer**

1. Open a command prompt and navigate to the *Samples\MobiLink\AutoScripting* subdirectory of your SQL Anywhere 9 installation.
2. Run the file *clean.bat*.

## Summary

During this tutorial, you:

- ◆ Created and populated Adaptive Server Anywhere consolidated and remote databases.
- ◆ Started a MobiLink synchronization server.
- ◆ Started the MobiLink synchronization client and synchronized the remote database with the consolidated database.

Learning  
accomplishments

During this tutorial, you gained:

- ◆ Familiarity with the MobiLink synchronization server and client as an integrated system.
- ◆ Competence in executing MobiLink synchronization server and client commands.
- ◆ Familiarity with the MobiLink synchronization server and client command lines and options.

---

## Further reading

The following documentation sections are good starting points for further reading.

☞ For more information about Adaptive Server Anywhere remote databases, see [“Adaptive Server Anywhere Clients”](#) [*MobiLink Clients*, page 59].

☞ For more information about running the MobiLink synchronization server, see [“Synchronization Basics”](#) [*MobiLink Administration Guide*, page 7].

☞ For more information about synchronization scripting, see [“Writing Synchronization Scripts”](#) [*MobiLink Administration Guide*, page 227].

☞ For a more advanced tutorial that describes how to write and add scripts, see [“Tutorial: Writing MobiLink Scripts and Monitoring Synchronizations”](#) on page 13.



---

## CHAPTER 2

# Tutorial: Writing MobiLink Scripts and Monitoring Synchronizations

About this chapter

This chapter provides a tutorial to guide you through the process of writing synchronization scripts, including conflict detection and resolution scripts. It also shows you how to monitor synchronizations.

Contents

<b>Topic:</b>	<b>page</b>
Introduction	14
Lesson 1: Set up the Adaptive Server Anywhere consolidated database	15
Lesson 2: Set up the remote Adaptive Server Anywhere databases	18
Lesson 3: Creating scripts for your synchronization	22
Lesson 4: Run MobiLink synchronization	25
Lesson 5: Monitoring your MobiLink synchronization using log files	26
Lesson 6: Creating scripts for conflict detection and resolution	28
Lesson 7: Use the MobiLink Monitor to detect update conflicts	31
Tutorial cleanup	37
Further reading	38

---

# Introduction

## Timing

The tutorial takes about 120 minutes.

## Goals

The goals for the tutorial are to gain competence and familiarity with the following tasks:

- ◆ Migrating the consolidated database schema to remote databases.
- ◆ Writing the basic scripts needed for synchronization and storing them in the consolidated database using Sybase Central or Interactive SQL.
- ◆ Writing scripts for conflict detection and resolution.
- ◆ Monitoring synchronization using log files and the MobiLink Monitor.

## Suggested background reading

- ☞ For more information about the architecture of MobiLink, see [“Synchronization Basics”](#) [*MobiLink Administration Guide*, page 7].
- ☞ For more information about synchronization scripts, see [“Introduction to synchronization scripts”](#) [*MobiLink Administration Guide*, page 228].
- ☞ For more information about Interactive SQL, see [“Using Interactive SQL”](#) [*Introducing SQL Anywhere Studio*, page 217].
- ☞ For more information about Sybase Central, see [“Managing Databases with Sybase Central”](#) [*Introducing SQL Anywhere Studio*, page 241].
- ☞ For an introduction to synchronization, see [“Tutorial: Introduction to MobiLink”](#) on page 1.
- ☞ For more information about MobiLink Conflict resolution, see [“Handling conflicts”](#) [*MobiLink Administration Guide*, page 64].
- ☞ For more information about the MobiLink Monitor, see [“MobiLink Monitor”](#) [*MobiLink Administration Guide*, page 117].

## Lesson 1: Set up the Adaptive Server Anywhere consolidated database

This lesson guides you through the following steps to set up your Adaptive Server Anywhere consolidated database:

1. Create the consolidated database and schema.
2. Define an ODBC data source for the consolidated database.

Create the consolidated database

In this procedure, you create the consolidated database using the Sybase Central Create a Database wizard.

### ❖ To create your Adaptive Server Anywhere RDBMS

1. Start Sybase Central.  
Choose Start ► Programs ► Sybase SQL Anywhere 9 ► Sybase Central.  
Sybase Central appears.
2. In Sybase Central, choose Tools ► Adaptive Server Anywhere 9 ► Create Database.  
The Create a Database wizard appears. Click Next.
3. Leave the default of Create a Database on this Computer. Click Next.
4. Enter the filename and path for the database. For example, enter:  
`C:\temp\cons.db`
5. Follow the remaining instructions in the wizard, accepting the default values, except as follows.
  - ◆ Choose a 4096 byte page size.  
A 4K page size increases performance for many environments.
6. Click Finish to connect to the new database in Sybase Central.

Generate the consolidated database schema

The consolidated database schema includes the Product table, storing the name and quantity of hardware products.

### ❖ To add the Product table to the consolidated schema

1. Start Interactive SQL.
  - ◆ In the left pane of Sybase Central, select the **cons - DBA** database. From the File menu choose Open Interactive SQL.  
Interactive SQL appears.

---

## 2. Install the Product table.

- ◆ Execute the following commands in Interactive SQL.

```
/* the Product table */
create table Product (
    name          varchar(128) not null primary key,
    quantity      integer,
    last_modified  timestamp default timestamp
)
go

insert into Product(name, quantity)
values ( 'Screwmaster Drill', 10);

insert into Product(name, quantity)
values ( 'Drywall Screws 10lb', 30);

insert into Product(name, quantity)
values ( 'Putty Knife x25', 12);

go
```

## 3. Install temporary tables used for conflict resolution.

In [“Lesson 6: Creating scripts for conflict detection and resolution”](#) on [page 28](#) you write scripts inserting values into these tables when a conflict occurs.

```
/* the Product_old table */
create table Product_old (
    name          varchar(128) not null primary key,
    quantity      integer,
    last_modified  timestamp default timestamp
)
go

/* the Product_new table */
create table Product_new (
    name          varchar(128) not null primary key,
    quantity      integer,
    last_modified  timestamp default timestamp
)
go
```

## 4. Verify the successful creation of each table.

- ◆ For example, execute the following command in Interactive SQL to verify the contents of the Product table.

```
select * from Product
```

Define an ODBC data source for the consolidated database.

Use the Adaptive Server Anywhere 9.0 driver to define an ODBC data source for the cons database.

❖ **To define an ODBC data source for the consolidated database**

1. Start the ODBC Administrator:  
From the Start menu, choose Programs ► SQL Anywhere 9 ► Adaptive Server Anywhere ► ODBC Administrator.  
The ODBC Data Source Administrator appears.
2. On the User DSN tab, click Add.  
The Create New Data Source dialog appears.
3. Select Adaptive Server Anywhere 9.0 and click Finish.  
The ODBC Configuration for Adaptive Server Anywhere 9 dialog appears.
4. On the ODBC tab, type the Data source name **asa\_cons**. On the Login tab, type **DBA** for the User ID and **SQL** for the Password. On the Database tab, type **cons** for the Server Name.
5. Click OK to add the data source.
6. Click OK to close the ODBC Administrator.

Further reading

- ☞ You can also use the dbinit command line utility to create your consolidated database. For more information, see [“Creating a database using the dbinit command-line utility”](#) [*ASA Database Administration Guide*, page 531] or [“Lesson 2: Set up the remote Adaptive Server Anywhere databases”](#) on page 18.
- ☞ For more information about consolidated databases, including non-ASA RDBMSs, see [“Consolidated database”](#) [*MobiLink Administration Guide*, page 10].
- ☞ For more information about Interactive SQL, see [“The Interactive SQL utility”](#) [*ASA Database Administration Guide*, page 538].
- ☞ For more information about creating ODBC data sources, see [“The Data Source utility”](#) [*ASA Database Administration Guide*, page 510].

---

## Lesson 2: Set up the remote Adaptive Server Anywhere databases

MobiLink is designed for synchronization involving a consolidated database server and a large number of mobile databases. In this section, you create two remote databases. For each database you:

- ◆ Migrate a selected portion of the consolidated schema.
- ◆ Create a synchronization publication, user, and subscription.

Create Adaptive Server Anywhere databases

In Lesson 1, you used Sybase Central to create databases. In this tutorial, you use a command line utility. The two tools produce identical results.

### ❖ To create and start Adaptive Server Anywhere remote databases

1. At a command prompt, navigate to the directory where you would like to create the remote databases.
2. Type the following commands to create the database:

```
dbinit -p 4096 remotel.db
```

For remote2, type:

```
dbinit -p 4096 remote2.db
```

The `-p` option defines a 4K page size shown to increase performance for many environments.

☞ For more information about `dbinit` options, see “[Creating a database using the `dbinit` command-line utility](#)” [*ASA Database Administration Guide*, page 531].

3. Now, to start the databases, type:

```
dbeng9 remotel.db
```

For remote2, type:

```
dbeng9 remote2.db
```

Migrating a subset of the consolidated database schema

Migrating a subset of the consolidated database schema involves:

- ◆ Connecting to the remote database.
- ◆ Creating a remote server and external login.
- ◆ Using the Sybase Central Data Migration wizard.

❖ **To migrate a subset of your consolidated database schema to remote1**

1. Start Sybase Central.

From the Start menu, choose Programs ► SQL Anywhere 9 ► Sybase Central.

2. Connect to the remote database:

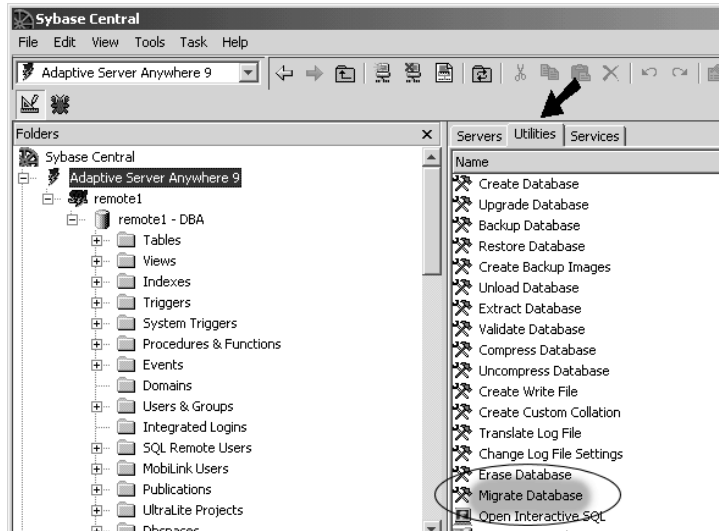
- ◆ Select Adaptive Server Anywhere 9 in the left pane of Sybase Central.
- ◆ From the File menu choose Connect.  
The Connect dialog appears.
- ◆ On the Identification tab, enter **DBA** as the User ID and **SQL** as the password. On the Database tab, enter **remote1** as the server name.
- ◆ Click OK to connect.

3. Create a remote server:

- ◆ Start the Remote Server Creation wizard.  
In the left pane, select the Remote Servers folder. From the File menu choose New ► Remote Server.  
The Remote Server Creation wizard starts.
- ◆ Name the remote server **my\_asa**. Click Next to continue.
- ◆ Choose Sybase Adaptive Server Anywhere as the type of server, and click Next.
- ◆ On the next page of the wizard, type the datasource name **asa\_cons** in the connection information section, and click Next.
- ◆ On the final page of the wizard, called “Make this a readonly data source”, choose Create an External Login. Use **DBA** for the Login name and **SQL** for the Password.
- ◆ Click Finish to exit the Remote Server Creation wizard.

4. Migrate the consolidated database schema:

- ◆ In the left pane, select the Adaptive Server Anywhere 9 plug-in. In the right pane, select the Utilities tab and double-click Migrate Database:



The Data Migration wizard starts.

- ◆ Select **remote1** as the destination database.
- ◆ On the next page, select **my\_asa** as the remote server. Click Next to continue.
- ◆ Choose **Product** as the only table to migrate, and click Next.
- ◆ Choose user DBA, and click Next.
- ◆ On the final page of the wizard, clear the Migrate the Data option.
- ◆ Click Finish.

5. Repeat the migration (steps 1 to 4 above) for the remote2 database.

Synchronization subscriptions and publications

Publications identify the tables and columns on your remote database that you want synchronized. These tables and columns are called **articles**. A synchronization subscription subscribes a MobiLink user to a publication.

Synchronization subscriptions and publications are stored in the remote database.

☞ For more information about defining publications and subscriptions, see [“Publishing data” \[MobiLink Clients, page 64\]](#).

### ❖ To create a remote synchronization publication, synchronization user, and synchronization subscription

1. Start Interactive SQL:

- ◆ In the left pane of Sybase Central, select the **remote1 - DBA** database. From the File menu choose Open Interactive SQL.



2. Enter synchronization information for remote1:

- ◆ Execute the following in Interactive SQL:

```
CREATE PUBLICATION pub_1 (TABLE Product);
CREATE SYNCHRONIZATION USER user_1;
CREATE SYNCHRONIZATION SUBSCRIPTION TO pub_1
  FOR user_1 TYPE TCPIP ADDRESS 'host=localhost'
  OPTION scriptversion='ver1';
```

3. Start Interactive SQL and connect to remote2.

4. Enter synchronization information for remote2:

- ◆ Execute the following in Interactive SQL:

```
CREATE PUBLICATION pub_2 (TABLE Product);
CREATE SYNCHRONIZATION USER user_2;
CREATE SYNCHRONIZATION SUBSCRIPTION TO pub_2
  FOR user_2 TYPE TCPIP ADDRESS 'host=localhost'
  OPTION scriptversion='ver1';
```

Now you have prepared the remote and consolidated databases. In the next lesson you write synchronization scripts. In Lesson 4 you run the synchronization.

---

## Lesson 3: Creating scripts for your synchronization

You can view, write, and modify synchronization scripts using Sybase Central. In this section you write the following synchronization scripts:

- ◆ **upload\_insert** To define how data inserted into the remote database is to be applied to the consolidated database.
- ◆ **download\_cursor** To define what data should be downloaded from the consolidated database.

Each script belongs to a designated **script version**. You must add a script version to the consolidated database before you add scripts.

### ❖ To add a script version

1. Connect to the cons database using the MobiLink plug-in in Sybase Central.
  - ◆ Select the MobiLink Synchronization 9 plug-in in the left pane of Sybase Central.
  - ◆ Choose Tools ► Connect.  
The New Connection dialog appears.
  - ◆ On the Identification tab, select the ODBC Data Source name option. Type **asa\_cons** for the data source name.
  - ◆ Click OK to connect.  
The **asa\_cons** datasource appears in the MobiLink plug-in.  
☞ For more information about the MobiLink plug-in, see “[MobiLink Help](#)” [*SQL Anywhere Studio Help*, page 173].
2. Add the script version **ver1**.  
In the left pane select the Versions folder. In the right pane double-click Add Version.  
The Add a New Script Version dialog appears.
3. Name the new version **ver1** and click Finish.

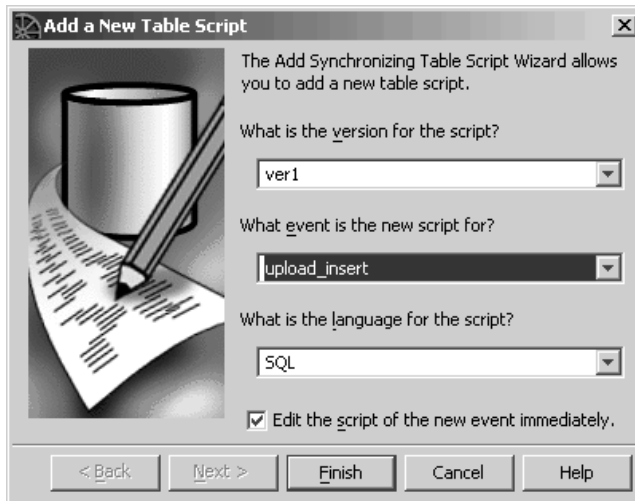
### ❖ To add synchronized tables to your consolidated database

1. In the MobiLink Synchronization plug-in of Sybase Central, open the Tables folder and double-click DBA.
2. Right-click the Product table and choose Add to Synchronized Tables.  
The Product table now appears in the Synchronized Tables folder.

Now that you have designated a synchronized table, you can add a new table script for each upload and download to the consolidated database.

❖ **To add table scripts for the Product table**

1. In the MobiLink Synchronization plug-in of Sybase Central, open the Synchronized Tables folder and select the Product table.
2. In the right pane double-click Add Table Script. The following dialog appears. Ensure that ver1 appears as the script version.



3. Select the **upload\_insert** event from the dropdown list and click Finish. The Product upload\_insert dialog appears.
4. Type the following SQL statement into the edit screen:

```
INSERT INTO Product( name, quantity, last_modified )  
VALUES( ?, ?, ? )
```

The upload\_insert event determines how data inserted into the remote database should be applied to the consolidated database.

☞ For more information about upload\_insert, see “[upload\\_insert table event](#)” [*MobiLink Administration Guide*, page 463]

5. Save the script.  
From the File menu, choose Save.
6. Repeat steps 1 to 5 for the **download\_cursor** event using the following SQL statement:

---

```
SELECT name, quantity, last_modified
FROM Product where last_modified >= ?
```

The `download_cursor` script defines a cursor to select consolidated database rows that are downloaded and inserted or updated in the remote database.

☞ For more information about `download_cursor`, see “[download\\_cursor table event](#)” [*MobiLink Administration Guide*, page 371]

#### Further reading

☞ For more information about the scripts you just created, see “[upload\\_insert table event](#)” [*MobiLink Administration Guide*, page 463] and “[download\\_cursor table event](#)” [*MobiLink Administration Guide*, page 371].

☞ For more information about script versions, see “[Script versions](#)” [*MobiLink Administration Guide*, page 239].

☞ For more information about adding scripts, see “[Adding and deleting scripts in your consolidated database](#)” [*MobiLink Administration Guide*, page 241].

☞ For more information about writing table scripts, see “[Table scripts](#)” [*MobiLink Administration Guide*, page 236].

☞ For more information about writing synchronization scripts, see “[Writing Synchronization Scripts](#)” [*MobiLink Administration Guide*, page 227].

☞ For a complete list of the events that you can program to customize your synchronization, see “[Synchronization Events](#)” [*MobiLink Administration Guide*, page 319].

## Lesson 4: Run MobiLink synchronization

### ❖ To run the synchronizations for remote1 and remote2

1. Start the MobiLink synchronization server.

At a command prompt, type the following on a single line:

```
dbmlsrv9 -c "dsn=asa_cons" -o mlserver.mls -v+ -dl -zu+ -x  
tcpip
```

☞ For a complete list of dbmlsrv9 options, see “[MobiLink Synchronization Server Options](#)” [*MobiLink Administration Guide*, page 189].

The MobiLink server window appears to indicate the MobiLink synchronization server is ready to handle requests.

2. Run the MobiLink synchronization client utility (dbmlsync) to initiate synchronizations.

At the command prompt, type the following on a single line:

```
dbmlsync -c "eng=remote1;uid=dba;pwd=sql" -o rem1.txt -v+
```

For remote2:

```
dbmlsync -c "eng=remote2;uid=dba;pwd=sql" -o rem2.txt -v+
```

☞ For a complete list of dbmlsync options, see “[Adaptive Server Anywhere Client Synchronization Parameters](#)” [*MobiLink Clients*, page 95].

Once you have started the MobiLink synchronization client, the DBMLSync window appears indicating that the MobiLink synchronization succeeded.

#### Further reading

☞ For more information about the MobiLink synchronization server, see “[The MobiLink synchronization server](#)” [*MobiLink Administration Guide*, page 11].

☞ For a complete list of dbmlsrv9 options, see “[MobiLink Synchronization Server Options](#)” [*MobiLink Administration Guide*, page 189].

☞ For more information about the dbmlsync, see “[Adaptive Server Anywhere Clients](#)” [*MobiLink Clients*, page 59].

☞ For a complete list of dbmlsync options, see “[Adaptive Server Anywhere Client Synchronization Parameters](#)” [*MobiLink Clients*, page 95].

---

## Lesson 5: Monitoring your MobiLink synchronization using log files

Once the tables have synchronized, you can view the progress of the synchronization using the two message log files you created with each command line, namely, *mlserver.mls*, *rem1.txt*, and *rem2.txt*. The default location of these files is the directory where the command was run.

### ❖ To find errors in a MobiLink synchronization log file

1. Open your log file in a text editor. For this tutorial, the log file is *mlserver.mls*.
2. Search the file for the string `MobiLink Server started`.
3. Scan down the left side of the file. A line beginning with *I.* contains an informational message, and a line beginning with *E.* contains an error message. For example:

```
I. 04/27 16:01:00. <Main>: ODBC isolation set to: Read Committed
I. 04/27 16:01:00. <Main>: Connected
I. 04/27 16:01:00. <Main>: Translated SQL:
                        SELECT COUNT(*) FROM ml_scripts_modified
E. 04/27 16:01:01. <Main>: Error: Unable to initialize communications stream 1: tcpip
I. 04/27 16:01:01. <Main>: Synchronization Server shutting down
I. 04/27 16:01:03. <Main>: Disconnected
I. 04/27 16:01:03. <Main>: Synchronization Server finished
```

4. Note that beside the *E.* in this example, there is the following text:

```
04/27 16:01:01. <Main>: Error: Unable to initialize
communications stream 1: tcpip.
```

This message indicates an error prior to the upload and download. There may be errors in the synchronization subscription or publication definitions.

5. Look for the clause that begins as follows:

```
ASA Synchronization request from:
```

This clause indicates that a synchronization request has been established.

6. Look for the clause that begins `Working on a request`. This indicates that the client and server are communicating. You may get this message if you have specified a high level of verbosity.

❖ **To detect errors in your MobiLink synchronization client log file**

1. Open the client log file *rem1.txt* in a text editor.
2. Search the file for the string `COMMIT`. If it appears, your synchronization was successful.
3. Search the file for the string `ROLLBACK`. If the transaction was rolled back, there were errors that prevented it from completing.
4. Scan down the left side of the file. If you see an *E.*, you have an error. If you don't have any errors, your synchronization has completed successfully.

Further reading

☞ For more information about MobiLink synchronization server log files, see “[Logging MobiLink synchronization server actions](#)” [*MobiLink Administration Guide*, page 12].

---

## Lesson 6: Creating scripts for conflict detection and resolution

Conflicts arise during the upload of rows to the consolidated database. If two users modify the same row on different remote databases, a conflict is detected when the second row arrives at the MobiLink synchronization server. Using synchronization scripts you can detect and resolve conflicts.

☞ For more information about MobiLink Conflict resolution, see “[Handling conflicts](#)” [*MobiLink Administration Guide*, page 64].

### Inventory example

Consider the scenario of two salesmen in the field. Salesman1 starts with an inventory of ten items, and then sells three. He updates the inventory on his remote database, remote1, to seven items. Salesman2 sells four items and updates her inventory (on remote2) to six.

When remote1 synchronizes using the MobiLink synchronization client utility the consolidated database is updated to seven. When remote2 synchronizes, a conflict is detected because the inventory value in the consolidated database has changed.

To resolve this conflict programmatically, you need three row values:

1. The current value in the consolidated database.  
After remote1 synchronizes, the value in the consolidated database is 7.
2. The new row value that Remote2 uploaded.
3. The old row value that Remote2 obtained during the previous synchronization.

In this case, you can use the following business logic to calculate the new inventory value and resolve the conflict:

```
current consolidated - (old remote - new remote)  
that is, 7 - (10-6) = 3
```

The expression (old remote - new remote) provides the number of items sold by Salesman2 rather than the absolute inventory value.

### Synchronization scripts for conflict detection and resolution

For conflict detection and resolution, you add the following scripts:

- ◆ **upload\_update** The `upload_update` event determines how data inserted into the remote database should be applied to the consolidated database. You can also use an extended prototype of `upload_update` to detect update conflicts.

For more information about using `upload_update` to detect conflicts, see “[Detecting conflicts](#)” [*MobiLink Administration Guide*, page 64].



☞ For more information about `upload_update`, see “[upload\\_update table event](#)” [*MobiLink Administration Guide*, page 475].

- ◆ **upload\_old\_row\_insert** You use this script to handle old row values obtained by the remote database during its previous synchronization.

☞ For more information about `upload_old_row_insert`, see “[upload\\_old\\_row\\_insert table event](#)” [*MobiLink Administration Guide*, page 467].

- ◆ **upload\_new\_row\_insert** You use this script to handle new row values (the updated values on the remote database).

☞ For more information about `upload_new_row_insert`, see “[upload\\_new\\_row\\_insert table event](#)” [*MobiLink Administration Guide*, page 465].

- ◆ **resolve\_conflict** The resolve conflict script applies business logic to resolve the conflict.

☞ For more information about `resolve_conflict`, see “[resolve\\_conflict table event](#)” [*MobiLink Administration Guide*, page 442].

## ❖ To install scripts for conflict detection and resolution

1. Start Interactive SQL.

- ◆ At a command prompt enter:

```
dbisql
```

The Connect dialog appears.

- ◆ On the identification tab, enter **DBA** as the User ID and **SQL** as the Password. On the Database tab, enter **cons** as the Server name.

2. Install the conflict detection and resolution scripts.

Execute the following in Interactive SQL:

```
/* upload_update */
call ml_add_table_script( 'ver1', 'Product',
'upload_update',
'UPDATE Product
  SET quantity = ?, last_modified = ?
  WHERE name = ?
  AND quantity=? AND last_modified=?' )
go

/* upload_old_row_insert */
call ml_add_table_script( 'ver1', 'Product',
'upload_old_row_insert',
'INSERT INTO Product_old (name,quantity,last_modified)
  values (?, ?, ?)' )
go
```

---

```

/* upload_new_row_insert */
call ml_add_table_script( 'ver1', 'Product',
  'upload_new_row_insert',
  'INSERT INTO Product_new (name,quantity,last_modified)
    values (?,?,?)' )
go

/* resolve_conflict */
call ml_add_table_script( 'ver1', 'Product',
  'resolve_conflict',
  'declare @product_name varchar(128);
  declare @old_rem_val integer;
  declare @new_rem_val integer;
  declare @curr_cons_val integer;
  declare @resolved_value integer;

  // obtain the product name
  SELECT name INTO @product_name
    FROM Product_old;

  // obtain the old remote value
  SELECT quantity INTO @old_rem_val
    FROM Product_old;

  //obtain the new remote value
  SELECT quantity INTO @new_rem_val
    FROM Product_new;

  // obtain the current value in cons
  SELECT quantity INTO @curr_cons_val
    FROM Product WHERE name = @product_name;

  // determine the resolved value
  SET @resolved_value =
    @curr_cons_val- (@old_rem_val - @new_rem_val);

  // update cons with the resolved value
  UPDATE Product
    SET quantity = @resolved_value
    WHERE name = @product_name;

  // clear the old and new row tables
  delete from Product_new;
  delete from Product_old
  ')

```

Setup for the Adaptive Server Anywhere consolidated database is complete.

#### Further reading

☞ For more information about MobiLink conflict detection and resolution, see [“Handling conflicts”](#) [*MobiLink Administration Guide*, page 64].

☞ For more information about MobiLink consolidated databases, see [“MobiLink Consolidated Databases”](#) [*MobiLink Administration Guide*, page 31].

## Lesson 7: Use the MobiLink Monitor to detect update conflicts

You can use the MobiLink Monitor to collect statistical information about synchronizations as they occur. The Monitor's graphical chart shows tasks on the vertical axis against the progression of time on the horizontal axis.

Using the Monitor, you can quickly identify synchronizations that result in errors or satisfy certain conditions. Since the Monitor does not significantly degrade performance, it is recommended for both development and production.

In this section you:

- ◆ Start and configure the MobiLink Monitor to visibly distinguish synchronizations that involve update conflicts.
- ◆ Generate a conflict by updating the same row on remote1 and remote2.
- ◆ Use the MobiLink Monitor to detect the conflict.

### ❖ To configure the MobiLink Monitor to detect update conflicts.

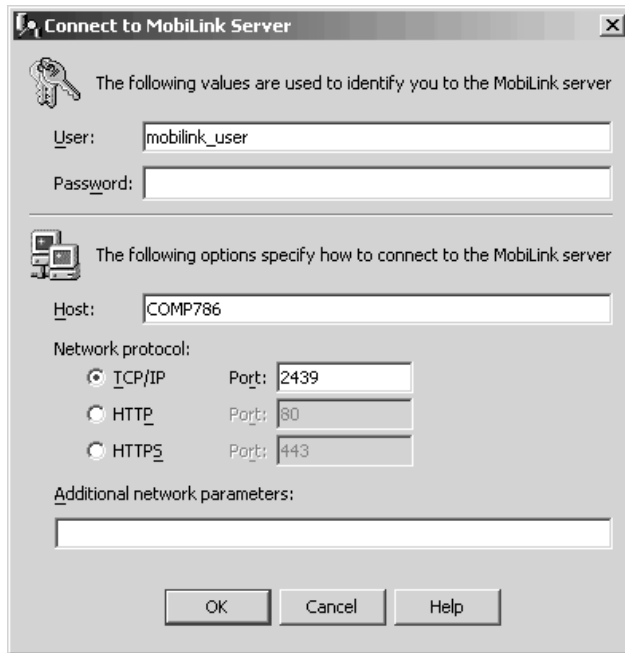
1. Start the MobiLink Monitor.

- ◆ From the Start menu, choose Programs ► SQL Anywhere 9 ► MobiLink ► MobiLink Monitor, or type the following at a command prompt:

```
dbmlmon
```

The Connect to MobiLink server dialog appears.

- ◆ Enter **monitor\_user** for the User ID. Since you started the MobiLink synchronization server with the `-zu+` option, this user will be added automatically. Leave the password field blank if you do not want to require a password for this user.



- ◆ Click OK to connect.

The MobiLink Monitor connects to the MobiLink synchronization server (dbmlsrv9).

2. Start the MobiLink Monitor Watch Manager.

From the MobiLink Monitor file menu choose Tools ► Watch Manager...

The Watch Manager dialog appears.

3. Add a new watch for update conflicts.

- ◆ Click New.

The New Watch dialog appears.

- ◆ Name the watch conflict\_detected.

- ◆ Choose **conflicted\_updates** for the Property field.

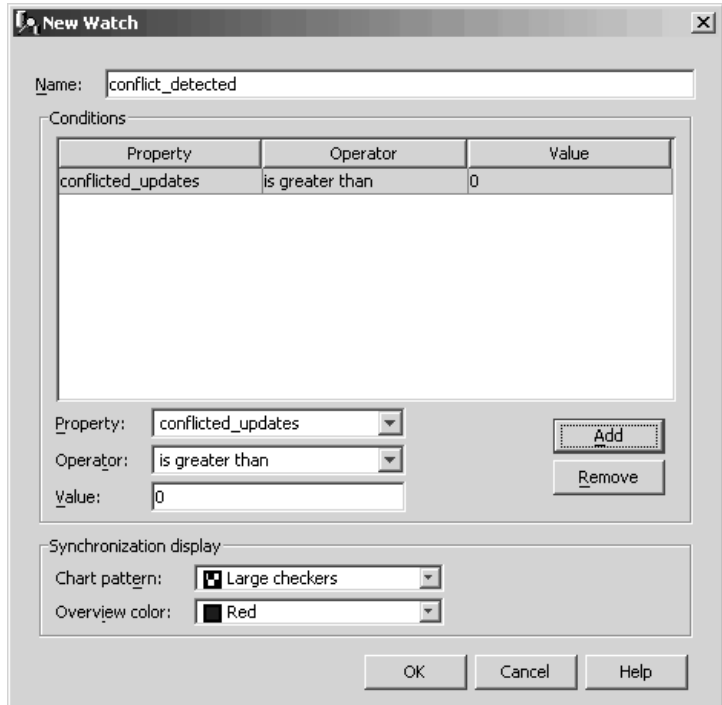
The conflicted\_updates statistical property indicates the number of uploaded updates for which conflicts were detected.

☞ For more information about MobiLink Monitor statistical properties, see “[MobiLink statistical properties](#)” [*MobiLink Administration Guide*, page 130].

- ◆ Set the watch to detect cases where one or more update conflicts occurred.

Set the Operator field to **is greater than**. Set the Value field to **0**.

- ◆ Click Add to save the settings.
- ◆ Select a pattern for the watch in the Chart pane. (The Chart pane is the middle pane in MobiLink Monitor.)
- ◆ Select a color for the watch in the Overview pane. (The Overview pane is the bottom pane in the MobiLink Monitor.)



4. Click OK to add the watch.

#### ❖ To generate an update conflict

1. Update the remote1 inventory value.

Salesman1 starts with a Screwmaster Drill inventory of ten items, and then sells three. He updates the inventory on his remote database, remote1, to seven items. To perform the update:

- ◆ Start Interactive SQL and connect to remote1 (if not already connected).

At a command prompt, type:

```
dbisql
```

The Connect dialog appears.

On the Identification tab, enter **DBA** as the User ID and **SQL** as the password. On the Database tab, enter **remote1** as the server name.

- 
- ◆ Update the Screwmaster Drill inventory to 7 items.

Execute the following in Interactive SQL.

```
UPDATE Product SET quantity = 7
WHERE name ='Screwmaster Drill'
COMMIT
```

2. Synchronize remote1.

At a command prompt, type the following to start the MobiLink synchronization client:

```
dbmlsync -c "eng=remote1;uid=dba;pwd=sql" -v+
```

Following synchronization, the consolidated database Screwmaster Drill inventory is 7 items.

3. Update the remote2 inventory value.

Salesman2 sells four items and updates her inventory (on remote2) to six. When remote2 synchronizes, a conflict is detected because the inventory value in the consolidated database has changed. To perform the update:

- ◆ Start Interactive SQL and connect to remote2.

At a command prompt, type:

```
dbisql
```

The Connect dialog appears.

On the Identification tab, enter **DBA** as the User ID and **SQL** as the password. On the Database tab, enter **remote2** as the server name.

- ◆ Update the Screwmaster Drill inventory to 6 items.

Execute the following in Interactive SQL.

```
UPDATE Product SET quantity = 6
WHERE name ='Screwmaster Drill'
COMMIT
```

4. Synchronize remote2.

At a command prompt, type the following to start the MobiLink synchronization client:

```
dbmlsync -c "eng=remote2;uid=dba;pwd=sql" -v+
```

Now you can switch to the MobiLink Monitor and view the results of the synchronization.

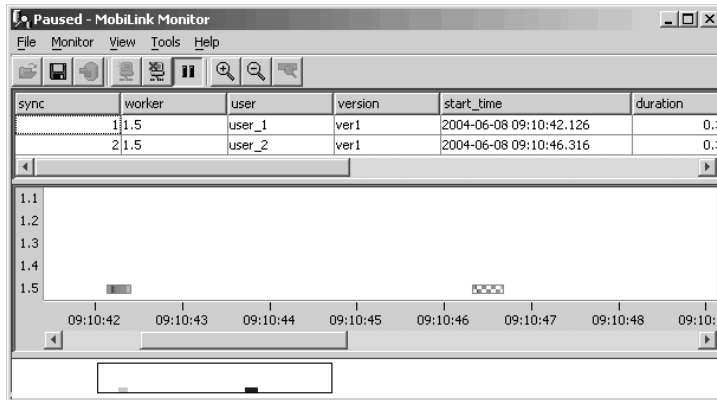
❖ **To detect the update conflict using the MobiLink Monitor**

1. Pause Chart Scrolling.

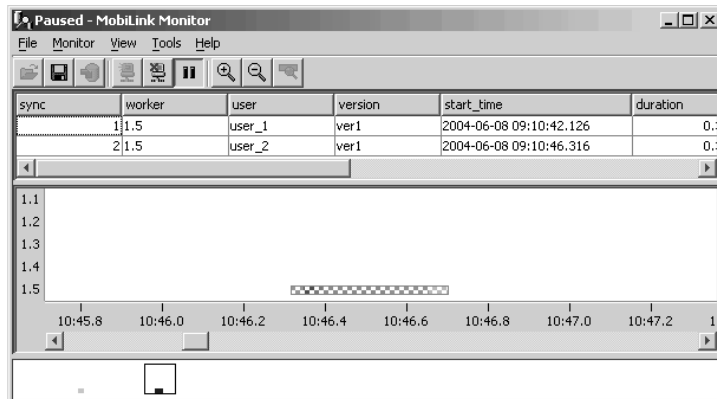
From the File menu, choose Monitor ► Pause Chart Scrolling.

2. View statistical information about the synchronization using the MobiLink Monitor's Overview pane, Chart pane, and Details table.

- ◆ Locate the synchronizations in the Monitor's Overview pane (the bottom pane in the MobiLink Monitor). The remote2 synchronization which generated an update conflict, appears in red:



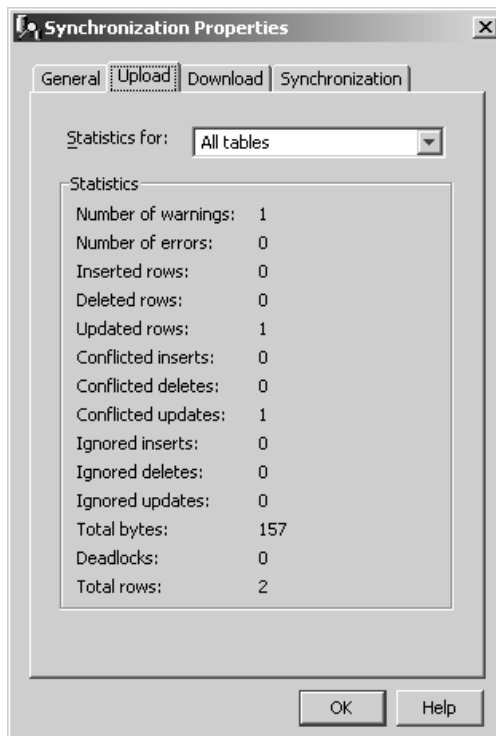
- ◆ To view the remote2 synchronization in the Chart pane, click and drag over the synchronization object in the Overview pane:



The synchronization object is displayed with the pattern you chose for the conflict\_detected watch.

- ◆ Use the zoom tool to view synchronization detail.  
From the File menu choose View ► Zoom In.

- ◆ To view synchronization properties, double-click the synchronization object or the corresponding row in the details table. Choose the Upload tab to see the number of conflicted updates.



#### Further reading

- ☞ For more information about MobiLink Conflict resolution, see [“Handling conflicts”](#) [*MobiLink Administration Guide*, page 64].
- ☞ For more information about the MobiLink Monitor, see [“MobiLink Monitor”](#) [*MobiLink Administration Guide*, page 117].
- ☞ For more information about MobiLink Monitor statistical properties, see [“MobiLink statistical properties”](#) [*MobiLink Administration Guide*, page 130].



## Tutorial cleanup

You should remove tutorial materials from your computer.

### ❖ To remove tutorial materials from your computer

1. Close all instances of the following applications:
  - ◆ The MobiLink Monitor
  - ◆ Sybase Central
  - ◆ Interactive SQL
2. Close the Adaptive Server Anywhere, MobiLink, and synchronization client windows by right-clicking each taskbar item and choosing Exit.
3. Delete all tutorial-related data sources.
  - ◆ Start ODBC Administrator.  
From the Start menu, choose Programs ► SQL Anywhere 9 ► Adaptive Server Anywhere ► ODBC Administrator.  
Select *asa\_cons* from the list of User Data Sources. Click Remove.
4. Delete the consolidated and remote databases.
  - ◆ Navigate to the directory containing your consolidated and remote databases.
  - ◆ Delete *cons.db*, *cons.log*, *remote1.db*, *remote1.log*, *remote2.db*, *remote2.log*.

---

## Further reading

The following documentation sections are a good starting point for further reading:

- ☞ For more information about running the MobiLink synchronization server, see [“The MobiLink synchronization server”](#) [*MobiLink Administration Guide*, page 11].
- ☞ For more information about synchronization scripting, see [“Writing Synchronization Scripts”](#) [*MobiLink Administration Guide*, page 227] and [“Synchronization Events”](#) [*MobiLink Administration Guide*, page 319].
- ☞ For an introduction to other methods of synchronization such as timestamp-based synchronization, see [“Synchronization Techniques”](#) [*MobiLink Administration Guide*, page 45].
- ☞ For information about testing your scripts in Sybase Central, see [“Testing script syntax”](#) [*MobiLink Administration Guide*, page 254].
- ☞ For more information about the MobiLink Monitor, see [“MobiLink Monitor”](#) [*MobiLink Administration Guide*, page 117].
- ☞ For more information about MobiLink conflict detection and resolution, see [“Handling conflicts”](#) [*MobiLink Administration Guide*, page 64].

---

## CHAPTER 3

# Tutorial: Using MobiLink with an Oracle 8i Consolidated Database

### About this chapter

This chapter provides a tutorial to guide you through the process of setting up a synchronization system when the consolidated database is an Oracle database and the remote database is an Adaptive Server Anywhere database.

This tutorial will show you how to set up and synchronize the two databases.

### Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Introduction</a>	40
<a href="#">Lesson 1: Create your databases</a>	41
<a href="#">Lesson 2: Running the MobiLink synchronization server</a>	47
<a href="#">Lesson 3: Running the MobiLink synchronization client</a>	48
<a href="#">Summary</a>	49
<a href="#">Further reading</a>	50

---

# Introduction

In this tutorial, you prepare an Oracle consolidated database and an Adaptive Server Anywhere remote database. You then synchronize the two databases using MobiLink.

## Required software

- ◆ A full SQL Anywhere Studio installation.
- ◆ A full installation of Oracle Enterprise Edition 8i.

## Competencies and experience

You should have the following competencies and experience before beginning the tutorial:

- ◆ Familiar with Sybase Central interface and functionality.
- ◆ Competent with Interactive SQL and Oracle SQL Plus.
- ◆ Competent programming Oracle.

## Goals

The goals for the tutorial are:

- ◆ To acquire familiarity with the MobiLink synchronization server and related components as they can be used with Oracle.
- ◆ To gain competence in executing MobiLink server and client commands as they pertain to an Oracle consolidated database.

## Suggested background reading

☞ For more information about writing SQL scripts, see [“Tutorial: Writing MobiLink Scripts and Monitoring Synchronizations”](#) on page 13.

☞ For more information about running the MobiLink synchronization server, see [“Synchronization Basics”](#) [*MobiLink Administration Guide*, page 7].

## Lesson 1: Create your databases

MobiLink synchronization requires that you have data a relational database, an ODBC data source for each database and two compatible databases.

### SQL files

You can enter data into an Oracle database using a number of different methods. This tutorial uses Oracle SQL Plus.

#### ❖ To create your databases

1. Start SQL Plus and connect to the Oracle consolidated database. Copy the following code into SQL Plus and execute it. These SQL statements drop, create and populate tables in the consolidated database. If there are no tables to drop, an error will appear in the SQL Plus output. This will not affect processing.

```
CREATE SEQUENCE emp_sequence;
CREATE SEQUENCE cust_sequence;
DROP TABLE emp;
CREATE TABLE emp ( emp_id int primary key, emp_name varchar(
    128 ) );
DROP TABLE cust;
CREATE TABLE cust ( cust_id int primary key, emp_id int
    references emp(emp_id), cust_name varchar( 128 ) );
INSERT INTO emp ( emp_id, emp_name ) VALUES ( emp_
    sequence.nextval, 'emp1' );
INSERT INTO emp ( emp_id, emp_name ) VALUES ( emp_
    sequence.nextval, 'emp2' );
INSERT INTO emp ( emp_id, emp_name ) VALUES ( emp_
    sequence.nextval, 'emp3' );
COMMIT;
INSERT INTO cust ( cust_id, emp_id, cust_name ) VALUES (
    cust_sequence.nextval, 1, 'cust1' );
INSERT INTO cust ( cust_id, emp_id, cust_name ) VALUES (
    cust_sequence.nextval, 1, 'cust2' );
INSERT INTO cust ( cust_id, emp_id, cust_name ) VALUES (
    cust_sequence.nextval, 2, 'cust3' );
COMMIT;
```

2. Start Interactive SQL.  
Choose Start ► Programs ► SQL Anywhere 9 ► Adaptive Server Anywhere ► Interactive SQL.
3. Connect to the remote database.
4. Copy the following code into Interactive SQL and execute it. These SQL statements drop and create tables in the remote databases. If there are no tables to drop, an error will appear in the Interactive SQL output. This

---

will not affect processing. Synchronization subscriptions and publications are also inserted to define the synchronization parameters for the MobiLink synchronization server.

```
CREATE TABLE emp ( emp_id int primary key ,emp_name varchar(
    128 ) );
CREATE TABLE cust ( cust_id int primary key, emp_id int
    references emp ( emp_id ), cust_name varchar( 128 )
    );
CREATE PUBLICATION emp_cust ( TABLE cust, TABLE emp );
CREATE SYNCHRONIZATION USER ml_user;
CREATE SYNCHRONIZATION SUBSCRIPTION
    TO emp_cust FOR ml_user TYPE TCPIP ADDRESS
    'host=localhost';
```

## ODBC data sources

You can now create ODBC data sources through which you connect to the Oracle consolidated database and the Adaptive Server Anywhere remote database. MobiLink requires an ODBC data source to perform data synchronization.

The consolidated data source

Ensure that you know your Instance, Service, and Database names, as these values are required for the ODBC portion of the installation. These values are established at the time of your Oracle installation.

The following steps set up an ODBC configuration for the Oracle consolidated database. You will set up the ODBC connections for the Adaptive Server Anywhere remote database later.

### ❖ To set up an ODBC data source for Oracle

1. Choose Start ► Programs ► Sybase SQL Anywhere 9 ► Adaptive Server Anywhere ► ODBC Administrator.  
The ODBC Data Source Administrator opens.
2. Click Add on the User DSN tab. The Create New Data Source window appears.
3. Select iAnywhere Solutions - Oracle 8, 8i and 9i Driver, and click Finish.  
The ODBC Oracle Driver Setup window appears.
4. Click the General tab and type the data source name **ora\_consol**. This is the DSN used for connecting to your Oracle database. You will need it later.
5. Enter the server name. This value depends on your Oracle installation. If the server is on your computer, you may be able to leave this field blank.

6. Click the Advanced tab. Enter a Default User Name. For this tutorial you can use **system**, or any User Name with sufficient rights to create objects. Click OK.
7. Click OK to close the ODBC Data Source Administrator.

## MobiLink system tables

MobiLink comes with a script called *syncora.sql*, located in the *MobiLink\setup* subdirectory of your SQL Anywhere installation. You run this script to set up your Oracle database to work with MobiLink.

*Syncora.sql* contains SQL statements, written in Oracle SQL, to prepare Oracle databases for use as consolidated databases. It creates a series of system tables, triggers, and procedures for use by MobiLink. The system tables are prefaced with *ML\_*. MobiLink works with these tables during the synchronization process.

### ❖ Create MobiLink system tables within Oracle

1. Start SQL Plus. Choose Start ► Programs ► Oracle - OraHome81 ► Application Development ► SQL Plus.

Connect to your Oracle database using Oracle SQL Plus. Log on using the **system** schema with password **manager**.

2. Run *syncora.sql* by typing the following command:

```
@path\syncora.sql;
```

where *path* is the *MobiLink\setup* subdirectory of your SQL Anywhere 9 installation. If there are spaces in your path, you should enclose the path and filename in quotation marks.

### ❖ To verify that the system tables are installed

1. Start SQL Plus. Choose Start ► Programs ► Oracle - OraHome81 ► Application Development ► SQL Plus.
2. Run the following SQL statement to yield a listing of the MobiLink system tables, procedures, and triggers:

```
SELECT object_name
FROM all_objects
WHERE object_name
LIKE 'ML_%';
```

If all of the objects shown in the following table are included, you can proceed to the next step.

---

## OBJECT\_NAME

---

ML\_ADD\_CONNECTION\_SCRIPT  
ML\_ADD\_DNET\_CONNECTION\_SCRIPT  
ML\_ADD\_DNET\_TABLE\_SCRIPT  
ML\_ADD\_JAVA\_CONNECTION\_SCRIPT  
ML\_ADD\_JAVA\_TABLE\_SCRIPT  
ML\_ADD\_LANG\_CONNECTION\_SCRIPT  
ML\_ADD\_LANG\_TABLE\_SCRIPT  
ML\_ADD\_TABLE\_SCRIPT  
ML\_ADD\_USER  
ML\_CONNECTION\_SCRIPT  
ML\_CONNECTION\_SCRIPT\_TRIGGER  
ML\_SCRIPT  
ML\_SCRIPTS\_MODIFIED  
ML\_SCRIPT\_TRIGGER  
ML\_SCRIPT\_VERSION  
ML\_SUBSCRIPTION  
ML\_TABLE  
ML\_TABLE\_SCRIPT  
ML\_TABLE\_SCRIPT\_TRIGGER  
ML\_USER

### Note

If any of the objects are missing, the procedure you just completed was not successful. In this case, you need to review the MobiLink error messages to see what went wrong; correct the problem; and then drop the MobiLink system tables as follows. However, do not drop system tables if there are any tables starting with ML\_ other than the ones listed above.

### ❖ To drop the MobiLink system tables

1. Run the following SQL statement in SQL Plus:

```
select 'drop ' || object_type || ' ' || object_name || ';'
from all_objects
where object_name like 'ML_%';
```

This generates a list of tables, procedures and triggers to be dropped.

2. Copy this list to a text file and save it as *drop.sql* in your *OracleTut* directory. Remove any lines that do not contain drop statements.



- Execute the SQL statements in *drop.sql* by running the following command:

```
@path\OracleTut\drop.sql;
```

Replace *path* with the location of your *OracleTut* directory. Run *drop.sql* a second time to delete tables that were not removed the first time because of dependencies.

- You can now repeat the instructions in Creating MobiLink system tables in Oracle, above.

The remote data source

#### ❖ To initialize your remote database

- Open a command prompt and navigate to your *OracleTut* directory; for example *c:\OracleTut*. Run the following command line:

```
dbinit remote.db
```

- Verify the successful creation of the database by getting a listing of the contents of this directory. The file *remote.db* should appear in the directory listing.

#### ❖ To create an ODBC data source for the remote database

- Open a command prompt and navigate to your *OracleTut* directory. Run the following command line:

```
dbdsn -w test_remote -y -c "uid=DBA;pwd=SQL;  
dbf=path\OracleTut\remote.db;eng=remote"
```

Replace *path* with the location of your *OracleTut* directory.

## Databases

In this procedure, you build a consolidated database using the Interactive SQL command line utility. The Interactive SQL utility helps you to execute SQL commands within your database. This procedure executes SQL statements within each database.

☞ For more information about Interactive SQL, see [“The Interactive SQL utility” \[ASA Database Administration Guide, page 538\]](#).

---

### ❖ To create and populate tables in the consolidated database

1. Start SQL Plus and connect to your consolidated database. Choose Start ► Programs ► Oracle - OraHome81 ► Application Development ► SQL Plus.
2. Execute the SQL statements in *build\_consol.sql* by running the following command:

```
@path\OracleTut\build_consol.sql;
```

Replace *path* with the location of your *OracleTut* directory. If the path contains spaces, enclose the path and filename in double quotes.

3. Verify the successful creation of each of the tables through SQL Plus directly from within the application. Run the following SQL statements:

```
SELECT * FROM emp;  
SELECT * FROM cust;
```

4. Leave the consolidated database running.

### ❖ To create tables and synchronization information in the remote database

1. Open a command prompt and navigate to your *OracleTut* directory. Run the following command line:

```
dbisql -c "dsn=test_remote" build_remote.sql
```

The Interactive SQL plug-in starts the remote database and executes the SQL statements in *build\_remote.sql*.

2. Verify the successful creation of the emp and cust tables using Interactive SQL or Sybase Central.
3. Leave the consolidated and remote databases running.

## Lesson 2: Running the MobiLink synchronization server

☞ The MobiLink synchronization server can now be started from a command prompt. Since MobiLink synchronization server is a client to the consolidated database, your consolidated database must be started prior to starting MobiLink. If you shut down your consolidated database following Lesson 1, you should restart the database.

### ❖ To start the MobiLink synchronization server

1. Ensure that your consolidated database is running.
2. Open a command prompt and navigate to your *OracleTut* directory. Run the following command line:

```
dbmlsrv9 -c "dsn=ora_consol;pwd=manager" -o mlserver.mls -v+  
-za -zu+
```

This command line specifies the following dbmlsrv9 options:

- ◆ **-c** Specifies connection parameters. Note that we only use the password as the User ID is contained in the DSN. For more information, see “**-c option**” [*MobiLink Administration Guide*, page 196].
- ◆ **-o** Specifies the message log file. For more information, see “**-o option**” [*MobiLink Administration Guide*, page 203].
- ◆ **-v+** Sets verbose logging on. For more information, see “**-v option**” [*MobiLink Administration Guide*, page 211].
- ◆ **-dl** Sets the display log feature ON.
- ◆ **-za** Turns automated scripting ON. For more information, see “**-za option**” [*MobiLink Administration Guide*, page 219].
- ◆ **-zu+** Automates the user authentication process. For more information, see “**-zu option**” [*MobiLink Administration Guide*, page 222].

### Further reading

☞ For more information about dbmlsrv9, see “**The MobiLink synchronization server**” [*MobiLink Administration Guide*, page 11] and “**MobiLink synchronization server**” [*MobiLink Administration Guide*, page 190].

---

## Lesson 3: Running the MobiLink synchronization client

The MobiLink client may now be started from a command prompt. The MobiLink client initiates synchronization.

You can specify connection parameters on the `dbmlsync` command line using the `-c` option. These parameters are for the *remote* database.

### ❖ To start the MobiLink client

1. Ensure that the MobiLink synchronization server is started.
2. Open a command prompt and navigate to your *OracleTut* directory. Run the following command line:

```
dbmlsync -c "dsn=test_remote" -o dbmlsync.out -v+ -e  
"SendColumnNames=ON"
```

This command line specifies the following `dbmlsync` options:

- ◆ **-c** Supply database connection parameters. For more information, see “[-c option](#)” [*MobiLink Clients*, page 102].
- ◆ **-o** Specify the message log file. For more information, see “[-o option](#)” [*MobiLink Clients*, page 142].
- ◆ **-v+** Verbose operation. For more information, see “[-v option](#)” [*MobiLink Clients*, page 150].
- ◆ **-e** Extended options. Specifying “`SendColumnNames=ON`” sends column names to MobiLink. For more information, see “[dbmlsync extended options](#)” [*MobiLink Clients*, page 105].

### Further reading

👉 For more information about `dbmlsync`, see “[MobiLink synchronization client](#)” [*MobiLink Clients*, page 96].

## Summary

During this tutorial, you accomplished the following tasks.

- ◆ Created a new Adaptive Server Anywhere database to serve as a remote database.
- ◆ Started a MobiLink synchronization server to work with your consolidated Oracle database.
- ◆ Started the MobiLink synchronization client and synchronized the remote database with the consolidated Oracle database.

Learning  
accomplishments

In this tutorial, you gained:

- ◆ Familiarity with the MobiLink synchronization server and client and how they work with an Oracle database.
- ◆ Competence in executing MobiLink server and client commands.

---

## Further reading

The following documentation areas are good starting points for further reading:

☞ For more information about running the MobiLink synchronization server, see [“Running the MobiLink synchronization server”](#) [*MobiLink Administration Guide*, page 11].

☞ For more information about synchronization scripting, see [“Writing Synchronization Scripts”](#) [*MobiLink Administration Guide*, page 227], and [“Synchronization Events”](#) [*MobiLink Administration Guide*, page 319].

☞ For an introduction to other methods of synchronization such as timestamp, see [“Synchronization Techniques”](#) [*MobiLink Administration Guide*, page 45].

---

## CHAPTER 4

# Tutorial: Java Synchronization Logic With Adaptive Server Anywhere

### About this chapter

This tutorial guides you through the basic steps for using Java synchronization logic. Using the CustDB sample as a consolidated database, you specify simple class methods for MobiLink table-level events. The process also involves running the MobiLink synchronization server (dbmlsrv9) with an option to set the path of compiled Java classes.

### Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Introduction</a>	52
<a href="#">Lesson 1: Compiling the CustdbScripts Java class</a>	53
<a href="#">Lesson 2: Specifying class methods for events</a>	55
<a href="#">Lesson 3: Run the MobiLink server with -sl java</a>	60
<a href="#">Lesson 4: Test synchronization</a>	61
<a href="#">Cleanup</a>	63
<a href="#">Further reading</a>	64

---

## Introduction

MobiLink connection-level and table-level event scripts can be written in SQL, Java or .NET. Java and .NET encapsulate event logic in class methods. In this tutorial you write Java class methods for MobiLink table-level events.

### Required software

- ◆ SQL Anywhere Studio 9.0.
- ◆ Java Software Development Kit.

### Competencies and experience

You will require:

- ◆ Familiarity with Java.
- ◆ Basic knowledge of MobiLink event scripts.

### Goals

You will gain competence and familiarity with:

- ◆ Utilizing simple Java class methods for MobiLink table-level events.

### Key concepts

This section uses the following steps to implement basic Java-based synchronization using the MobiLink CustDB sample database:

- ◆ Compiling the *CustdbScripts.class* file with MobiLink API references.
- ◆ Specifying class methods for particular table-level events.
- ◆ Running the MobiLink server (dbmlsrv9) with the -sl java option.
- ◆ Testing synchronization with a sample Windows client application.

### Suggested background reading

☞ For more information about synchronization scripts, see [“Introduction to synchronization scripts”](#) [*MobiLink Administration Guide*, page 228].

☞ For more information about Sybase Central, see [“Managing Databases with Sybase Central”](#) [*Introducing SQL Anywhere Studio*, page 241].



## Lesson 1: Compiling the CustdbScripts Java class

Java classes encapsulate synchronization logic in methods.

In this lesson, you will compile a class associated with the CustDB sample database.

MobiLink Database  
Sample

SQL Anywhere Studio ships with an Adaptive Server Anywhere sample database (CustDB) that is already set up for synchronization, including the SQL scripts required to drive synchronization. The CustDB ULCustomer table, for example, is a synchronized table supporting a variety of table-level events.

CustDB is designed to be a consolidated database server for both UltraLite and Adaptive Server Anywhere clients. The CustDB database has a DSN called UltraLite 9.0 Sample.

### The CustdbScripts class

In this section, you create a Java class called CustdbScripts with logic to handle the ULCustomer upload\_insert and upload\_update events. You enter the CustdbScripts code in a text editor and save the file as *CustdbScripts.java*.

#### ❖ To create CustdbScripts.java

1. Create a directory for the Java class and assembly.

This tutorial assumes the path *c:\mljava*.

2. Using a text editor, enter the CustdbScripts code:

```
public class CustdbScripts
{
    public static String UploadInsert()
    {
        return("INSERT INTO ULCustomer(cust_id,cust_name) values
            (?,?)");
    }
    public String DownloadCursor(java.sql.Timestamp ts,String
        user )
    {
        return("SELECT cust_id, cust_name FROM ULCustomer where
            last_modified >= ' " + ts + " ' ");
    }
}
```

**Note:**

The class and associated methods must be set as public.

- 
3. Save the file as *CustdbScripts.java* in *c:\mljava*.

## Compiling the Java class

### The MobiLink API

To execute Java synchronization logic, the MobiLink synchronization server must have access to the classes in *mlscript.jar*. This jar file contains a repository of MobiLink API classes to utilize in your Java methods.

☞ For more information about the MobiLink Java API, see “[MobiLink Java API Reference](#)” [*MobiLink Administration Guide*, page 273].

When compiling a Java class for MobiLink, you must include this JAR file to make use of the API. In this section, you use the javac utility `-classpath` option to specify *mlscript.jar* for the *CustdbScripts* class.

### ❖ To compile the Java class (Windows)

1. On the command line, navigate to the folder containing *CustdbScripts.java* (*c:\mljava*).
2. Enter the following:

```
javac custdbscripts.java -classpath "%asany9%\java\mlscript.jar"
```

The *CustdbScripts.class* file is generated.

### Further reading

☞ For more information about the MobiLink API, see “[MobiLink Java API Reference](#)” [*MobiLink Administration Guide*, page 273].

☞ For more information about Java methods, see “[Methods](#)” [*MobiLink Administration Guide*, page 261].

☞ For more information about the CustDB sample database, and using alternate RDBMS servers, see “[Setting up the CustDB consolidated database](#)” on page 102.

## Lesson 2: Specifying class methods for events

*CustdbScripts.class*, created in the previous lesson, encapsulates the methods `UploadInsert()` and `DownloadCursor()`. These methods contain implementation for the `ULCustomer` `upload_insert` and `upload_update` events, respectively.

In this section, you will specify class methods for table-level events using two approaches:

1. Using the MobiLink Synchronization plug-in.

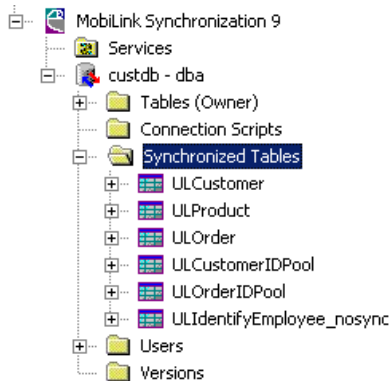
You connect to the CustDB database with Sybase Central, change the language for the `upload_insert` script to Java, and specify `CustdbScripts.UploadInsert` to handle the event.

2. Using the `ml_add_java_table_script` stored procedure.

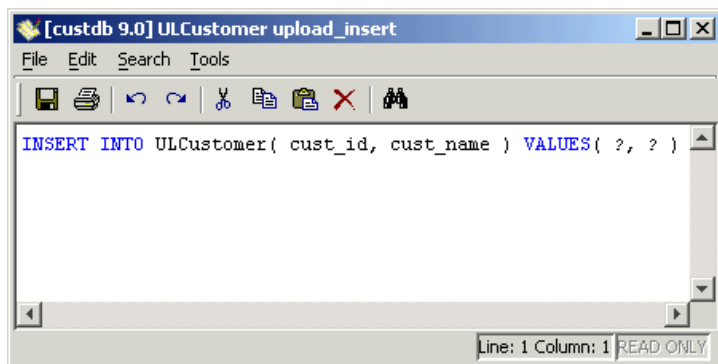
You will connect to the CustDB database with Interactive SQL and execute `ml_add_java_table_script`, specifying `CustdbScripts.DownloadCursor` for the `download_cursor` event.

### ❖ To specify `CustdbScripts.UploadInsert` for the `ULCustomer` `upload_insert` event

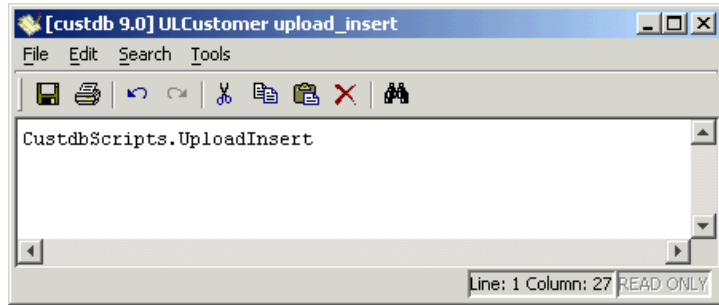
1. Connect to the sample database using the MobiLink Synchronization plug-in:
  - ◆ Start Sybase Central.
  - ◆ In the left pane, right-click the MobiLink Synchronization 9 plug-in and choose Connect.
  - ◆ On the Identification tab, choose UltraLite 9.0 Sample for the ODBC Data Source name.  
On the Database tab, ensure the option to search for network database servers is not selected.
  - ◆ Click OK to Connect.
  - ◆ Sybase Central should now display the CustDB data source under the MobiLink Synchronization 9 plug-in.



2. Change the language for the ULCustomer table upload\_insert event to Java:
  - ◆ In the left pane, open the Synchronized Tables folder and select the ULCustomer table. A list of table-level scripts appears in the right pane.
  - ◆ Click on the table script associated with the custdb 9.0 upload\_insert event. From the File menu, choose Set Script Language to Java.
3. Enter the fully-qualified Java method name as the upload\_insert script.
  - ◆ Double-click the table script associated with the upload\_insert event. A window revealing the script contents appears:



- ◆ Change the script contents to the fully-qualified method name, **CustdbScripts.UploadInsert**:



**Note:**

The fully qualified method name is case sensitive.

- ◆ To save the script, select Save from the File menu.

4. Exit Sybase Central.

This step used Sybase Central to specify a Java method as the script for the ULCustomer upload\_insert event.

Alternatively, you can use the ml\_add\_java\_connection\_script and ml\_add\_java\_table\_script stored procedures. Using these stored procedures is more efficient, especially if you have a large number of Java methods to handle synchronization events.

☞ For more information, see “ml\_add\_java\_connection\_script” [*MobiLink Administration Guide*, page 483] and “ml\_add\_java\_table\_script” [*MobiLink Administration Guide*, page 484].

In the next section you will learn how to add a Java method as the script for a MobiLink event using the ml\_add\_java\_connection\_script stored procedure or the ml\_add\_java\_table\_script stored procedure.

To do this, you will connect to CustDB with Interactive SQL and execute ml\_add\_java\_table\_script, assigning CustdbScripts.DownloadCursor to the download\_cursor event.

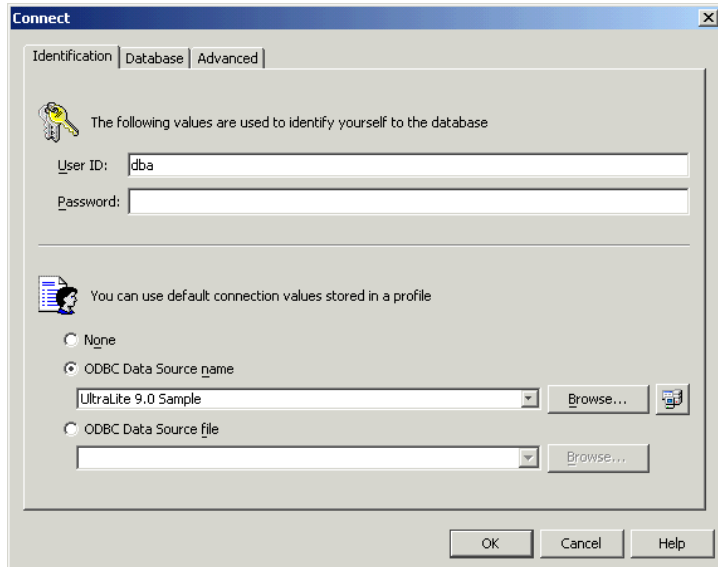
❖ **To specify CustdbScripts.DownloadCursor for the ULCustomer download\_cursor event**

1. Connect to the sample database with Interactive SQL.

- ◆ Open Interactive SQL.

The Connect dialog appears.

- ◆ On the Identification tab, choose UltraLite 9.0 Sample for the ODBC Data Source:



On the Database tab, ensure the option to search for network database servers is not selected.

2. Execute the following command in Interactive SQL:

```
call ml_add_java_table_script(
'custdb 9.0',
'ULCustomer',
'download_cursor',
'CustdbScripts.DownloadCursor');
commit;
```

Following is a description of each parameter:

Parameter	Description
custdb 9.0	The script version.
ULCustomer	The synchronized table.
download_cursor	The event name.
CustdbScripts.DownloadCursor	The fully qualified Java method.

3. Exit Interactive SQL.

In this lesson, you specified your Java methods to handle ULCustomer table-level events. The next lesson ensures that the MobiLink server loads the appropriate class files and the MobiLink API.

Further reading

☞ For more information about adding scripts with stored procedures, see [“ml\\_add\\_java\\_connection\\_script”](#) [*MobiLink Administration Guide*, page 483]

and “ml\_add\_java\_table\_script” [*MobiLink Administration Guide*, page 484].

☞ For general information about adding and deleting synchronization scripts, see “Adding and deleting scripts in your consolidated database” [*MobiLink Administration Guide*, page 241].

---

## Lesson 3: Run the MobiLink server with -sl java

Running the MobiLink server with the `-sl java -cp` option specifies a set of directories to search for class files, and forces the Java Virtual Machine to load on server startup.

### ❖ To start the MobiLink server (dbmlsrv9) and load Java assemblies

1. On a command line, enter the following on a single line:

```
dbmlsrv9 -c "dsn=ultralite 9.0 sample" -sl java (-cp c:\mljava)
```

A message dialog appears indicating that the server is ready to handle requests. Now the Java method is executed when the ULCustomer table `upload_insert` event triggers during synchronization.

#### Further reading

☞ For more information, see “[-sl dnet option](#)” [*MobiLink Administration Guide*, page 207].



## Lesson 4: Test synchronization

UltraLite comes with a sample Windows client that automatically invokes the dbmlsync utility when the user issues a synchronization. It is a simple sales-status application that you can run against the CustDB consolidated database you started in the previous lesson.

### Start the application (Windows)

#### ❖ To start and synchronize the sample application

1. Launch the sample application.

From the Start menu, choose Programs ► Sybase SQL Anywhere 9 ► UltraLite ► Windows Sample Application.

2. Enter an employee ID.

Input a value of **50** and press Enter.

The application automatically synchronizes, and a set of customers, products, and orders are downloaded to the application from the CustDB consolidated database.

In the next section you will enter a new customer name and order details. During a subsequent synchronization, this information will be uploaded to the CustDB consolidated database and the upload\_insert and download\_cursor events for the ULCustomer table will trigger.

### Add an order (Windows)

#### ❖ To add an order

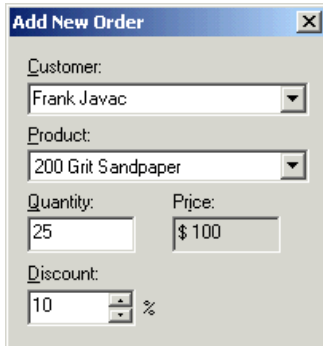
1. From the Order menu, choose New.

The Add New Order screen is displayed.

2. Enter a new Customer Name.

For example, enter **Frank Javac**.

3. Choose a product, and enter the quantity and discount:



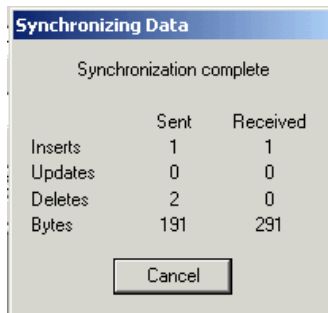
4. Press Enter to add the new order.

You have now modified the data in your local UltraLite database. This data is not shared with the consolidated database until you synchronize.

❖ **To synchronize with the consolidated database and trigger the upload\_insert event**

1. From the file menu, choose Synchronize.

A window appears indicating one Insert successfully uploaded to the consolidated database.



Further reading

👉 For more information about the CustDB Windows Application, see [“The CustDB Sample Application” on page 99](#).

## Cleanup

You should remove tutorial materials from your computer.

### ❖ To remove tutorial materials from your computer

1. Return the ULCustomer table upload\_insert and download\_cursor scripts to their original SQL logic.

- ◆ Open Interactive SQL.

The Connect dialog appears.

- ◆ On the Identification tab, choose UltraLite 9.0 Sample for the ODBC Data Source.
- ◆ Click OK to Connect.
- ◆ Execute the following commands in Interactive SQL:

```
call ml_add_table_script( 'custdb 9.0',
    'ULCustomer',
    'upload_insert',
    'INSERT INTO ULCustomer( cust_id, cust_name ) VALUES(
        ?, ? )' );
```

```
call ml_add_table_script( 'custdb 9.0',
    'ULCustomer',
    'download_cursor',
    'SELECT "cust_id", "cust_name" FROM "ULCustomer"
    WHERE "last_modified" >= ?' );
```

2. Close the Adaptive Server Anywhere, MobiLink, and synchronization client windows by right-clicking on each taskbar item and choosing Close.
3. Delete all tutorial-related Java sources.

Delete the folder containing your *CustdbScripts.java* and *CustdbScripts.class* files (*c:\mljava*).

**Note:**

Ensure that you do not have other important files in *c:\mljava*.

---

## Further reading

The following documentation sections are a good starting point for further reading:

☞ For more information about writing MobiLink synchronization scripts in Java, see [“Setting up Java synchronization logic”](#) [*MobiLink Administration Guide*, page 257].

☞ For an example illustrating the use of Java synchronization scripts for custom authentication, see [“Java synchronization example”](#) [*MobiLink Administration Guide*, page 267].

☞ For more information about synchronization scripting, see [“Writing Synchronization Scripts”](#) [*MobiLink Administration Guide*, page 227] and [“Synchronization Events”](#) [*MobiLink Administration Guide*, page 319].

☞ For an introduction to other methods of synchronization such as timestamp, see [“Synchronization Techniques”](#) [*MobiLink Administration Guide*, page 45].

---

## CHAPTER 5

# Tutorial: .NET Synchronization Logic With Adaptive Server Anywhere

### About this chapter

This tutorial guides you through the basic steps for using .NET synchronization logic. Using the CustDB sample as a consolidated database, you specify simple class methods for MobiLink table-level events. The process also involves running the MobiLink synchronization server (dbmlsrv9) with an option that sets the path of .NET assemblies.

### Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Introduction</a>	66
<a href="#">Lesson 1: Compile the CustdbScripts.dll assembly with MobiLink references</a>	67
<a href="#">Lesson 2: Specify class methods for events</a>	72
<a href="#">Lesson 3: Run MobiLink with -sl dnet</a>	77
<a href="#">Lesson 4: Test synchronization</a>	78
<a href="#">Cleanup</a>	80
<a href="#">Further reading</a>	81

---

## Introduction

MobiLink connection-level and table-level event scripts can be written in SQL, Java, or .NET. Both Java and .NET encapsulate event logic in class methods. In this tutorial you will subscribe .NET class methods to MobiLink table-level events.

### Required software

- ◆ SQL Anywhere Studio 9.0.
- ◆ Microsoft .NET Framework SDK.

### Competencies and experience

You will require:

- ◆ Familiarity with .NET.
- ◆ Basic knowledge of MobiLink event scripts.

### Goals

You will gain competence and familiarity with:

- ◆ Utilizing .NET class methods for MobiLink table-level event scripts.

### Key concepts

This section uses the following steps to implement basic .NET synchronization using the MobiLink CustDB sample database:

- ◆ Compiling the *CustdbScripts.dll* private assembly with MobiLink references.
- ◆ Specifying class methods for table-level events.
- ◆ Running the MobiLink server (dbmlsrv9) with the `-sl dnet` option.
- ◆ Testing synchronization with a sample Windows client application.

### Suggested background reading

☞ For more information about synchronization scripts, see [“Introduction to synchronization scripts”](#) [*MobiLink Administration Guide*, page 228].

☞ For more information about Sybase Central, see [“Managing Databases with Sybase Central”](#) [*Introducing SQL Anywhere Studio*, page 241].

## Lesson 1: Compile the CustdbScripts.dll assembly with MobiLink references

.NET classes encapsulate synchronization logic in methods.

In this lesson, you will compile a class associated with the CustDB sample database.

MobiLink Database Sample

SQL Anywhere Studio ships with an Adaptive Server Anywhere sample database (CustDB) that is already set up for synchronization, including the SQL scripts required to drive synchronization. The CustDB ULCustomer table, for example, is a synchronized table supporting a variety of table-level events.

CustDB is designed to be a consolidated database server for both UltraLite and Adaptive Server Anywhere clients. The CustDB database has a DSN called UltraLite 9.0 Sample.

In this section, you create a .NET class called CustdbScripts with logic to handle the ULCustomer upload\_insert and upload\_update events.

### The CustdbScripts assembly

The MobiLink API

To execute .NET synchronization logic, the MobiLink synchronization server must have access to the classes in *iAnywhere.MobiLink.Script.dll*. *iAnywhere.MobiLink.Script.dll* contains a repository of MobiLink .NET API classes to utilize in your .NET methods.

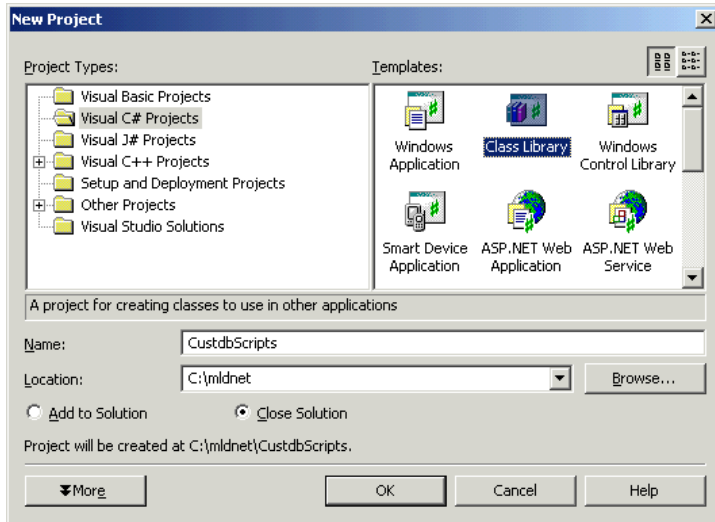
☞ For more information about the MobiLink .NET API, see “[MobiLink .NET API Reference](#)” [*MobiLink Administration Guide*, page 303].

When compiling the CustdbScripts class, you must include this assembly to make use of the API. You can compile your class using Visual Studio .NET or from the command line.

- ◆ In Visual Studio .NET, create a new Class Library and enter the CustdbScripts code. Link *iAnywhere.MobiLink.Script.dll*, and build the assembly for your class.
- ◆ On the command line, enter the CustdbScripts code in a text editor and save the file as *CustdbScripts.cs* ( *CustdbScripts.vb* for Visual Basic .NET). Using a command line compiler, reference *iAnywhere.MobiLink.Script.dll* and build the assembly for your class.

❖ **To create the CustdbScripts assembly using Visual Studio .NET**

1. Start a new Visual C# or Visual Basic .NET Class Library project.  
Use CustdbScripts for the project Name and enter an appropriate path.  
This tutorial assumes the path `c:\mldnet`.



2. Enter the CustdbScripts code.

For C#, type:

```
namespace MLEExample
{
    class CustdbScripts
    {
        public static string UploadInsert()
        {
            return("INSERT INTO ULCustomer(cust_id,cust_name) values
                (?,?)");
        }
        public static string DownloadCursor(System.DateTime ts,
            string user )
        {
            return("SELECT cust_id, cust_name FROM ULCustomer WHERE
                last_modified >= ' " + ts.ToString("yyyy-MM-dd
                hh:mm:ss.fff") + "'");
        }
    }
}
```

For Visual Basic .NET, type:



```
Namespace MLExample

Class CustdbScripts

    Public Shared Function UploadInsert() As String
        Return("INSERT INTO ULCustomer(cust_id,cust_name) values
            (?,?)")
    End Function

    Public Shared Function DownloadCursor(ByVal ts As
        System.DateTime, ByVal user As String) As String
        Return("SELECT cust_id, cust_name FROM ULCustomer " + _
            "WHERE last_modified >= '" + ts.ToString("yyyy-MM-dd
            hh:mm:ss.fff") + "'")
    End Function

End Class

End Namespace
```

3. Add a reference to the MobiLink API.

- ◆ From the Visual Studio .NET Project menu, choose Add Existing Item...
- ◆ Select *iAnywhere.MobiLink.Script.dll* in the *win32* directory of your SQL Anywhere Studio installation. From the Open drop down menu choose Link File.

4. Build *CustdbScripts.dll*.

From the Build menu choose Build CustdbScripts.

This creates *CustdbScripts.dll* in

*C:\mldnet\CustdbScripts\CustdbScripts\bin\Debug*.

---

❖ **To create the CustdbScripts assembly using the command line**

1. Create a directory for the .NET class and assembly.

This tutorial assumes the path `c:\mldnet`.

2. Using a text editor, enter the CustdbScripts code.

For C#, type:

```
namespace MLExample
{
class CustdbScripts
{
    public static string UploadInsert()
    {
        return("INSERT INTO ulcustomer(cust_id,cust_name) values
            (?,?)");
    }
    public static string DownloadCursor(System.DateTime ts,
        string user )
    {
        return("SELECT cust_id, cust_name FROM ULCustomer where
            last_modified >= ' " + ts.ToString("yyyy-MM-dd
            hh:mm:ss.fff") + "'");
    }
}
}
```

For Visual Basic .NET, type:

```
Namespace MLExample

Class CustdbScripts

    Public Shared Function UploadInsert() As String
        Return("INSERT INTO ULCustomer(cust_id,cust_name) values
            (?,?)")
    End Function

    Public Shared Function DownloadCursor(ByVal ts As
        System.DateTime, ByVal user As String) As String
        Return("SELECT cust_id, cust_name FROM ULCustomer " + _
            "WHERE last_modified >= ' " + ts.ToString("yyyy-MM-dd
            hh:mm:ss.fff") + "'")
    End Function

End Class

End Namespace
```

3. Save the file as `CustdbScripts.cs` ( `CustdbScripts.vb` for Visual Basic .NET) in `c:\mldnet`.
4. Compile the file using the following command.

For C#, type:

```
csc /out:c:\mldnet\custdbscripts.dll /target:library  
/reference:"%asany9%\win32\  
iAnywhere.MobiLink.Script.dll" c:\mldnet\  
CustdbScripts.cs
```

For Visual Basic .NET, type:

```
vbc /out:c:\mldnet\custdbscripts.dll /target:library  
/reference:"%asany9%\win32\  
iAnywhere.MobiLink.Script.dll" c:\mldnet\  
CustdbScripts.vb
```

The *CustdbScripts.dll* assembly is generated.

Further reading

- ☞ For more information about the MobiLink API, see “[MobiLink .NET API Reference](#)” [*MobiLink Administration Guide*, page 303].
- ☞ For more information about .NET methods, see “[Methods](#)” [*MobiLink Administration Guide*, page 288].

---

## Lesson 2: Specify class methods for events

☞ For more information about the CustDB sample database, and using alternate RDBMS servers, see “[Setting up the CustDB consolidated database](#)” on page 102.

*CustdbScripts.dll*, created in the previous lesson, encapsulates the methods UploadInsert() and DownloadCursor(). These methods contain implementation for the ULCustomer upload\_insert and upload\_update events, respectively.

In this section, you specify class methods for table-level events using two approaches:

1. Using the MobiLink Synchronization plug-in.

You connect to the CustDB database with Sybase Central, change the language for the upload\_insert script to .NET, and specify MLExample.CustdbScripts.UploadInsert to handle the event.

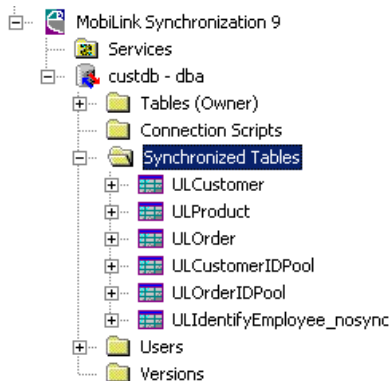
2. Using the ml\_add\_dnet\_table\_script stored procedure.

You will connect to the CustDB database with Interactive SQL and execute ml\_add\_dnet\_table\_script, specifying MLExample.CustdbScripts.DownloadCursor for the download\_cursor event.

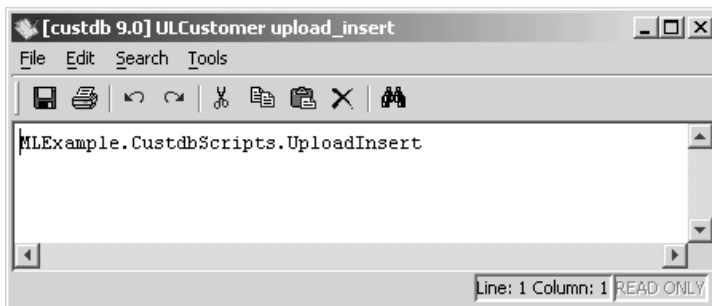
### ❖ To subscribe CustdbScripts.uploadInsert() to the upload\_insert event for the ULCustomer table

1. Connect to the sample database using the MobiLink Synchronization plug-in:

- ◆ Start Sybase Central.  
Choose Start ► Programs ► Sybase SQL Anywhere 9 ► Sybase Central.
- ◆ In the left pane, right-click the MobiLink Synchronization 9 plug-in and choose Connect.
- ◆ Choose the UltraLite 9.0 Sample as the ODBC Data Source.  
On the Identification tab, select the ODBC Data Source name option. Type **UltraLite 9.0 Sample** for the data source name.  
On the Database tab, ensure the option to search for network database servers is not selected.
- ◆ Click OK to Connect.
- ◆ Sybase Central should now display the CustDB data source under the MobiLink Synchronization 9 plug-in.



2. Change the language for the ULCustomer table upload\_insert event to .NET:
  - ◆ In the left pane, open the Synchronized Tables folder and select the ULCustomer table. A list of table-level scripts appears in the right pane.
  - ◆ Click on the table script associated with the custdb 9.0 upload\_insert event. From the File menu, choose Set Script Language to .NET.
3. Enter the fully-qualified .NET method name for the upload\_insert script.
  - ◆ Double-click the table script associated with the upload\_insert event. A window revealing the script contents appears.
  - ◆ Change the script contents to the fully-qualified method name, **MLExample.CustdbScripts.UploadInsert**.



**Note:**

The fully qualified method name is case sensitive.

- ◆ To save the script, choose Save from the File menu.
4. Exit Sybase Central.

---

This step used Sybase Central to specify a .NET method as the script for the ULCustomer upload\_insert event.

Alternatively, you can use the ml\_add\_dnet\_connection\_script and ml\_add\_dnet\_table\_script stored procedures. Using these stored procedures is more efficient, especially if you have a large number of .NET methods to handle synchronization events.

☞ For more information, see “ml\_add\_dnet\_connection\_script” [*MobiLink Administration Guide*, page 482] and “ml\_add\_dnet\_table\_script” [*MobiLink Administration Guide*, page 483].

In the next section you connect to CustDB with Interactive SQL and execute ml\_add\_dnet\_table\_script, assigning MLExample.CustdbScripts.DownloadCursor to the download\_cursor event.

❖ **To specify MLExample.CustdbScripts.DownloadCursor for the ULCustomer download\_cursor event**

1. Connect to the sample database with Interactive SQL.

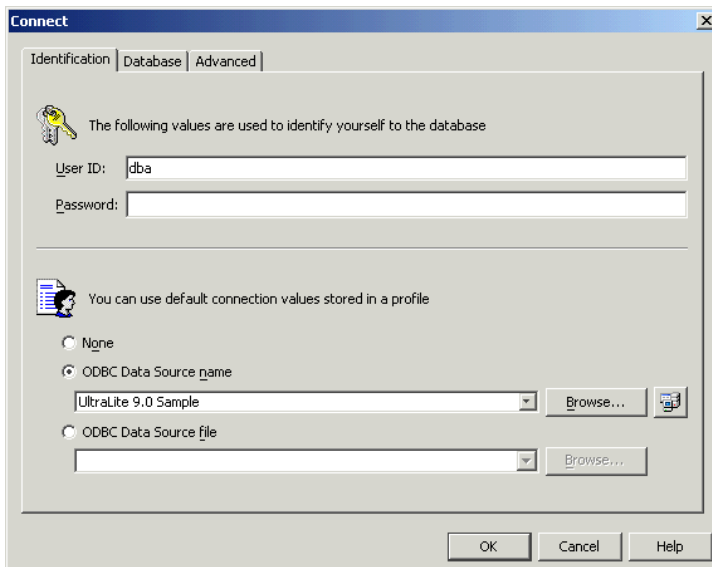
- ◆ Start Interactive SQL.

Choose Start ► Programs ► SQL Anywhere 9 ► Adaptive Server Anywhere ► Interactive SQL, or enter the following at a command prompt:

```
dbisql
```

The Connect dialog appears.

- ◆ On the Identification tab, choose UltraLite 9.0 Sample for the ODBC Data Source:



On the Database tab, ensure the option to search for network database servers is not selected.

2. Execute the following command in Interactive SQL:

```
call ml_add_dnet_table_script(
'custdb 9.0',
'ULCustomer',
'download_cursor',
'MLExample.CustdbScripts.DownloadCursor');
commit;
```

Following is a description of each parameter:

Parameter	Description
custdb 9.0	The script version.
ULCustomer	The synchronized table.
download_cursor	The event name.
MLExample,CustdbScripts.- DownloadCursor	The fully qualified .NET method.

3. Exit Interactive SQL.

In this lesson, you specified .NET methods to handle ULCustomer table-level events. The next lesson ensures that the MobiLink server loads the appropriate class files and the MobiLink API.

---

## Further reading

☞ For more information about adding and deleting synchronization scripts, see “Adding and deleting scripts in your consolidated database” [*MobiLink Administration Guide*, page 241].

☞ For more information about the scripts used in this lesson, see “ml\_add\_dnet\_connection\_script” [*MobiLink Administration Guide*, page 482] and “ml\_add\_dnet\_table\_script” [*MobiLink Administration Guide*, page 483].



## Lesson 3: Run MobiLink with -sl dnet

Running the MobiLink server with the `-sl dnet` option specifies the location of .NET assemblies and forces the CLR to load on server startup.

If you used Visual Studio .NET to compile, the location of *CustdbScripts.dll* is `c:\mldnet\CustdbScripts\CustdbScripts\bin\Debug`. If you used the command line, the location of *CustdbScripts.dll* is `c:\mldnet`.

### ❖ To start the MobiLink server (dbmlsrv9) and load .NET assemblies

1. Start the MobiLink server with the `-sl dnet` option.

If you used Visual Studio .NET to compile your assembly:

On a command line, enter the following on a single line:

```
dbmlsrv9 -c "dsn=ultralite 9.0 sample" -dl -o cons1.txt -v+
-sl dnet (-MLAutoLoadPath=c:\mldnet\CustdbScripts\
CustdbScripts\bin\Debug)
```

If you used the command line to compile your assembly:

On a command line, enter the following on a single line:

```
dbmlsrv9 -c "dsn=ultralite 9.0 sample" -dl -o cons1.txt -v+
-sl dnet (-MLAutoLoadPath=c:\mldnet)
```

A message dialog appears indicating that the server is ready to handle requests. Now the .NET method is executed when the `upload_insert` events triggers during synchronization.

#### Further reading

For more information see, “`-sl dnet` option” [*MobiLink Administration Guide*, page 207].

---

## Lesson 4: Test synchronization

UltraLite comes with a sample Windows client that automatically invokes the dbmlsync utility when the user issues a synchronization. It is a simple sales-status application that you can run against the CustDB consolidated database you started in the previous lesson.

### Start the application (Windows)

#### ❖ To start and synchronize the sample application

1. Launch the sample application.

From the Start menu, choose Programs ► SQL Anywhere 9 ► UltraLite ► Windows Sample Application.

2. Enter an employee ID.

Input a value of **50** and press ENTER.

The application automatically synchronizes, and a set of customers, products, and orders are downloaded to the application from the CustDB consolidated database.

In the next section you will enter a new customer name and order details. During a subsequent synchronization, this information will be uploaded to the CustDB consolidated database and the upload\_insert and download\_cursor events for the ULCustomer table will trigger.

### Add an order (Windows)

#### ❖ To add an order

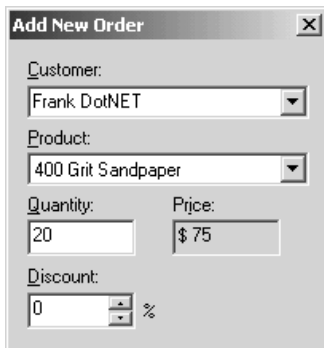
1. From the Order menu, choose New.

The Add New Order screen appears.

2. Enter a new Customer Name.

For example, enter Frank DotNET.

3. Choose a product, and enter the quantity and discount:



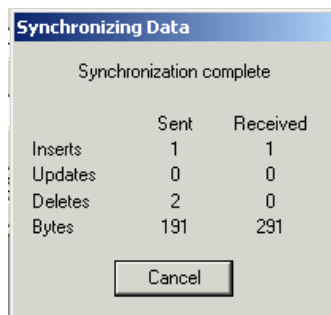
4. Press Enter to add the new order.

You have now modified the data in your local UltraLite database. This data is not shared with the consolidated database until you synchronize.

❖ **To synchronize with the consolidated database and trigger the `upload_insert` event**

1. From the file menu, choose Synchronize.

A window appears indicating one Insert successfully uploaded to the consolidated database.



Further reading

➔ For more information about the CustDB Windows Application, see [“The CustDB Sample Application” on page 99](#).

---

## Cleanup

You should remove tutorial materials from your computer.

### ❖ To remove tutorial materials from your computer

1. Return the ULCustomer table upload\_insert and download\_cursor scripts to their original SQL logic.

- ◆ Open Interactive SQL.

Choose Start ► Programs ► SQL Anywhere 9 ► Adaptive Server Anywhere ► Interactive SQL, or enter the following at a command prompt:

```
dbisql
```

The Connect dialog appears.

- ◆ On the Identification tab, choose UltraLite 9.0 Sample for the ODBC Data Source.
- ◆ Click OK to Connect.
- ◆ Execute the following commands in Interactive SQL:

```
call ml_add_table_script( 'custdb 9.0',  
    'ULCustomer',  
    'upload_insert',  
    'INSERT INTO ULCustomer( cust_id, cust_name ) VALUES(  
        ?, ? )' );
```

```
call ml_add_table_script( 'custdb 9.0',  
    'ULCustomer',  
    'download_cursor',  
    'SELECT "cust_id", "cust_name" FROM "ULCustomer"  
    WHERE "last_modified" >= ?' );
```

2. Close the Adaptive Server Anywhere, MobiLink, and synchronization client windows by right-clicking on each taskbar item and choosing Close.
3. Delete all tutorial-related .NET sources.

Delete the folder containing your *CustdbScripts.cs* and *CustdbScripts.dll* files (*c:\mldnet*).

**Note:**

Ensure that you do not have other important files in *c:\mldnet*.

## Further reading

The following documentation sections are a good starting point for further reading:

☞ For more information about writing MobiLink synchronization scripts in .NET, see [“Setting up .NET synchronization logic”](#) [*MobiLink Administration Guide*, page 283].

☞ For information about debugging .NET synchronization logic, see [“Debugging .NET synchronization logic”](#) [*MobiLink Administration Guide*, page 292].

☞ For a detailed example illustrating the use of .NET synchronization scripts for custom authentication, see [“.NET synchronization example”](#) [*MobiLink Administration Guide*, page 300].

☞ For more information about synchronization scripting, see [“Writing Synchronization Scripts”](#) [*MobiLink Administration Guide*, page 227] and [“Synchronization Events”](#) [*MobiLink Administration Guide*, page 319].

☞ For an introduction to other methods of synchronization such as timestamp, see [“Synchronization Techniques”](#) [*MobiLink Administration Guide*, page 45].



---

## CHAPTER 6

# The Contact Sample Application

### About this chapter

This chapter uses the sample application called Contact to illustrate a variety of techniques that you can use for common synchronization tasks.

The techniques are illustrated using SQL scripts. Many of the same techniques can be implemented using Java or .NET synchronization logic.

### Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Introduction</a>	84
<a href="#">Setup</a>	85
<a href="#">Tables in the Contact databases</a>	87
<a href="#">Users in the Contact sample</a>	90
<a href="#">Synchronization</a>	91
<a href="#">Monitoring statistics and errors in the Contact sample</a>	98

---

## Introduction

This chapter introduces you to the Contact sample application. This sample is a valuable resource for the MobiLink developer. It provides you with an example of how to implement many of the techniques you will need to develop MobiLink applications.

The Contact sample application includes an Adaptive Server Anywhere consolidated database and two Adaptive Server Anywhere remote databases. It illustrates several common synchronization techniques. To get the most out of this chapter, you should study the sample application as you read.

Although the consolidated database is an Adaptive Server Anywhere database, the synchronization scripts consist of SQL statements that should work with minimal changes on other database management systems.

The Contact sample is in the *Samples\MobiLink>Contact* subdirectory of your SQL Anywhere installation. For an overview, see *Samples\MobiLink>Contact\readme.txt*.

### Synchronization design

The synchronization design in the Contact sample application uses the following features:

- ◆ **Column subsets** A subset of the columns of the Customer, Product, SalesRep, and Contact tables on the consolidated database are shared with the remote databases.
- ◆ **Row subsets** All of the columns but only one of the rows of the SalesRep table on the consolidated database are shared with each remote database.
  - ☞ For more information, see “[Partitioning rows among remote databases](#)” [*MobiLink Administration Guide*, page 52].
- ◆ **Timestamp-based synchronization** This is a way of identifying changes that were made to the consolidated database since the last time a device synchronized. The Customer, Contact, and Product tables are synchronized using a method based on timestamps.
  - ☞ For more information, see “[Timestamp-based synchronization](#)” [*MobiLink Administration Guide*, page 48].



## Setup

A Windows batch file called *build.bat* is provided to build the Contact sample databases. On UNIX systems, the file is *build.sh*. You may want to examine the contents of the batch file. It carries out the following actions:

- ◆ Creates ODBC data source definitions for a consolidated database and each of two remote databases.
- ◆ Creates a consolidated database named *consol.db* and loads the MobiLink system tables, database schema, some data, synchronization scripts, and MobiLink user names into the database.
- ◆ Creates two remote databases, each named *remote.db*, in subdirectories named *remote\_1* and *remote\_2*. Loads information common to both databases and applies customizations. These customizations include a global database identifier, a MobiLink user name, and subscriptions to two publications.

### ❖ To build the Contact sample

1. Open a command prompt and navigate to the *Samples\MobiLink>Contact* subdirectory of your SQL Anywhere installation.
2. Run *build.bat* (Windows) or *build.sh* (Unix).

## Running the Contact sample

The Contact sample includes batch files that carry out initial synchronizations and illustrate MobiLink synchronization server and *dbmlsync* command lines. You can examine the contents of the following batch files, located in the *Samples\MobiLink>Contact* subdirectory of your SQL Anywhere 9 installation, in a text editor:

- ◆ *step1.bat*
- ◆ *step2.bat*
- ◆ *step3.bat*

### ❖ To run the Contact sample

1. Start the MobiLink synchronization server.
  - ◆ Open a command prompt and navigate to the *Samples\MobiLink>Contact* subdirectory of your SQL Anywhere 9 installation and execute the following command:

---

`step1`

This command runs a batch file that starts the MobiLink synchronization server in a verbose mode. This mode is useful during development or troubleshooting, but has a significant performance impact and so would not be used in a routine production environment.

2. Synchronize both remote databases.

- ◆ Open a command prompt and navigate to the *Samples\MobiLink>Contact* subdirectory of your SQL Anywhere 9 installation and execute the following command:

`step2`

This is a batch file that synchronizes both remote databases.

3. Shut down the MobiLink synchronization server.

- ◆ Open a command prompt and navigate to the *Samples\MobiLink>Contact* subdirectory of your SQL Anywhere 9 installation and execute the following command:

`step3`

This is a batch file that shuts down the MobiLink synchronization server.

To explore how synchronization works in the Contact sample, you can use Interactive SQL to modify the data in the remote and consolidated databases, and use the batch files to synchronize.

## Tables in the Contact databases

The table definitions for the Contact database are located in the following files:

- ◆ *Samples\MobiLink>Contact\build\_consol.sql*
- ◆ *Samples\MobiLink>Contact\build\_remote.sql*

Both the consolidated and the remote databases contain the following three tables, although their definition is slightly different in each place.

### SalesRep

Each sales representative occupies one row in the SalesRep table. Each remote database belongs to a single sales representative.

In each remote database, SalesRep has the following columns:

- ◆ **rep\_id** A primary key column that contains an identifying number for the sales representative.
- ◆ **name** The name of the representative.

In the consolidated database only, there is also an ml\_username column holding the MobiLink user name for the representative.

### Customer

This table holds one row for each customer. Each customer is a company with which a single sales representative does business. There is a one-to-many relationship between the SalesRep and Customer tables.

In each remote database, Customer has the following columns:

- ◆ **cust\_id** A primary key column holding an identifying number for the customer.
- ◆ **name** The customer name. This is a company name.
- ◆ **rep\_id** A foreign key column referencing the SalesRep table. Identifies the sales representative assigned to the customer.

In the consolidated database, there are two additional columns, last\_modified and active:

- ◆ **last\_modified** The last time the row was modified. This column is used for timestamp-based synchronization.
- ◆ **active** A BIT column that indicates if the customer is currently active (1) or if the company no longer deals with this customer (0). If the column is marked inactive (0) all rows corresponding to this customer are deleted from remote databases.

---

## Contact

This table holds one row for each contact. A contact is a person who works at a customer company. There is a one-to-many relationship between the Customer and Contact tables.

In each remote database, Contact has the following columns:

- ◆ **contact\_id** A primary key column holding an identifying number for the customer.
- ◆ **name** The name of the individual contact.
- ◆ **cust\_id** The identifier of the customer for whom the contact works.

In the consolidated database, the table also has the following columns:

- ◆ **last\_modified** The last time the row was modified. This column is used for timestamp-based synchronization.
- ◆ **active** A BIT column that indicates if the contact is currently active (1) or if the company no longer deals with this contact (0). If the column is marked inactive (0) the row corresponding to this contact is deleted from remote databases.

## Product

Each product sold by the company occupies one row in the Product table. The Product table is held in a separate publication so that remote databases can synchronize the table separately.

In each remote database, Product has the following columns:

- ◆ **id** A primary key column holding an identifying number for the product.
- ◆ **name** The name of the individual item.
- ◆ **size** The size of the item.
- ◆ **quantity** The number of items in stock. When a sales representative takes an order, this column is updated.
- ◆ **unit\_price** The price per unit of the product.

In the consolidated database, the Product table has the following additional columns:

- ◆ **supplier** The company that manufactures the product.
- ◆ **last\_modified** The last time the row was modified. This column is used for timestamp-based synchronization.
- ◆ **active** A BIT column that indicates if the contact is currently active (1) or if the company no longer deals with this contact (0). If the column is marked inactive (0), the row corresponding to this contact is deleted from remote databases.

In addition to these tables, a set of tables is created at the consolidated database only. These include the `product_conflict` table, which is a temporary table used during conflict resolution, and a set of tables for monitoring MobiLink activities owned by a user named `mlmaint`. Scripts to create the MobiLink monitoring tables are in the file *Samples\MobiLink>Contact\mlmaint.sql*.

---

## Users in the Contact sample

Several different database user IDs and MobiLink user names are included in the Contact sample.

### Database user IDs

The two remote databases belong to sales representatives Samuel Singer (rep\_id 856) and Pamela Savarino (rep\_id 949).

When connecting to their remote database, these users both use the default Adaptive Server Anywhere user ID **dba** and password **SQL**.

Each remote database also has a user ID **sync\_user** with password **sync\_user**. This user ID is employed only on the dbmlsync command line. It is a user with REMOTE DBA authority, and so can carry out any operation when connected from dbmlsync, but has no authority when connected from any other application. The widespread availability of the user ID and password is thus not a problem.

At the consolidated database, there is a user named **mlmaint**, who owns the tables used for monitoring MobiLink synchronization statistics and errors. This user has no right to connect. The assignment of the tables to a separate user ID is done simply to separate the objects from the others in the schema for easier administration in Sybase Central and other utilities.

### MobiLink user names

MobiLink user names are distinct from database user IDs. Each remote device has a MobiLink user name in addition to the user ID they use when connecting to a database. The MobiLink user name for Samuel Singer is SSinger. The MobiLink user name for Pamela Savarino is PSavarino. The MobiLink user name is stored or used in the following locations:

- ◆ At the remote database, the MobiLink user name is added using a `CREATE SYNCHRONIZATION USER` statement.
- ◆ At the consolidated database, the MobiLink user name and password are added using the dbmluser utility.
- ◆ During synchronization, the MobiLink password for the connecting user is supplied on the dbmlsync command line listed in *Samples\MobiLink\Contact\step2.bat*.
- ◆ The MobiLink synchronization server supplies the MobiLink user name as a parameter to many of the scripts during synchronization.
- ◆ The SalesRep table at the consolidated database has an ml\_username column. The synchronization scripts match the MobiLink user name parameter against the value in this column.

## Synchronization

The following sections describe the Contact sample's synchronization logic.

### Synchronizing sales representatives in the Contact sample

The synchronization scripts for the SalesRep table illustrates **snapshot synchronization**. Regardless of whether a sales representative's information has changed, it is downloaded.

☞ For more information, see “[Snapshot synchronization](#)” [*MobiLink Administration Guide*, page 50].

#### Business rules

The business rules for the SalesRep table are as follows:

- ◆ The table must not be modified at the remote database.
- ◆ A sales representative's MobiLink user name and rep\_id value must not change.
- ◆ Each remote database contains a single row from the SalesRep table, corresponding to the remote database owner's MobiLink user name.

#### Downloads

- ◆ **download\_cursor** At each remote database, the SalesRep table contains a single row. There is very little overhead for the download of a single row, so a simple snapshot **download\_cursor** script is used:

```
SELECT rep_id, name
FROM SalesRep
WHERE ? IS NOT NULL
AND ml_username = ?
```

The first parameter in the script is the last download timestamp, which is not used. The IS NOT NULL expression is a dummy expression supplied to use the parameter. The second parameter is the MobiLink user name.

#### Uploads

This table should not be updated at the remote database, so there are no upload scripts for the table.

### Synchronizing customers in the Contact sample

The synchronization scripts for the Customer table illustrate **timestamp-based synchronization** and partitioning rows. Both of these techniques minimize the amount of data that is transferred during synchronization while maintaining consistent table data.

☞ For more information, see “[Timestamp-based synchronization](#)” [*MobiLink Administration Guide*, page 48].

---

☞ For more information, see “[Partitioning rows among remote databases](#)” [*MobiLink Administration Guide*, page 52].

## Business rules

The business rules governing customers are as follows:

- ◆ Customer information can be modified at both the consolidated and remote databases.
- ◆ Periodically, customers may be reassigned among sales representatives. This process is commonly called territory realignment.
- ◆ Each remote database contains only the customers they are assigned to.

## Downloads

- ◆ **download\_cursor** The following **download\_cursor** script downloads only active customers for whom information has changed since the last successful download. It also downloads only customers assigned to a particular sales representative.

```
SELECT cust_id, Customer.name, Customer.rep_id
FROM Customer key join SalesRep
WHERE Customer.last_modified >= ?
AND SalesRep.ml_username = ?
AND Customer.active = 1
```

- ◆ **download\_delete\_cursor** The following **download\_delete\_cursor** script downloads only customers for whom information has changed since the last successful download. It deletes all customers marked as inactive or who are not assigned to the sales representative.

```
SELECT cust_id
FROM Customer key join SalesRep
WHERE Customer.last_modified >= ?
AND ( SalesRep.ml_username != ? OR Customer.active = 0 )
```

If rows are deleted from the Customer table at the consolidated database, they do not appear in this result set and so are not deleted from remote databases. Instead, customers are marked as inactive.

When territories are realigned, this script deletes those customers no longer assigned to the sales representative. It also deletes customers who are transferred to other sales representatives. Such additional deletes are flagged with a SQLCODE of 100 but do not interfere with synchronization. A more complex script could be developed to identify only those customers transferred away from the current sales representative.

The MobiLink client carries out cascading deletes at the remote database, so this script also deletes all contacts who work for customers assigned to some other sales representative.



## Uploads

Customer information can be inserted, updated, or deleted at the remote database. The scripts corresponding to these operations are as follows:

- ◆ **upload\_insert** The following **upload\_insert** script adds a row to the Customer table, marking the customer as active:

```
INSERT INTO Customer(
    cust_id, name, rep_id, active )
VALUES ( ?, ?, ?, 1 )
```

- ◆ **upload\_update** The following **upload\_update** script modifies the customer information at the consolidated database:

```
UPDATE Customer
SET name = ?, rep_id = ?
WHERE cust_id = ?
```

Conflict detection is not carried out on this table.

- ◆ **upload\_delete** The following **upload\_delete** script marks the customer as inactive at the consolidated database. It does not delete a row.

```
UPDATE Customer
SET active = 0
WHERE cust_id = ?
```

## Synchronizing contacts in the Contact sample

The Contact table contains the name of a person working at a customer company, a foreign key to the customer and a unique integer identifying the contact. It also contains a last\_modified timestamp and a marker to indicate whether the contact is active.

## Business rules

The business rules for this table are as follows:

- ◆ Contact information can be modified at both the consolidated and remote databases.
- ◆ Each remote database contains only those contacts who work for customers they are assigned to.
- ◆ When customers are reassigned among sales representatives, contacts must also be reassigned

## Trigger

A trigger on the Customer table is used to ensure that the contacts get picked up when information about a customer is changed. The trigger explicitly alters the last\_modified column of each contact whenever the corresponding customer is altered:

---

```

CREATE TRIGGER UpdateCustomerForContact
AFTER UPDATE OF rep_id ORDER 1
ON DBA.Customer
REFERENCING OLD AS old_cust NEW as new_cust
FOR EACH ROW
BEGIN
    UPDATE Contact
    SET Contact.last_modified = new_cust.last_modified
    FROM Contact
    WHERE Contact.cust_id = new_cust.cust_id
END

```

By updating all contact records whenever a customer is modified, the trigger ties the customer and their associated contacts together so that whenever a customer is modified, all associated contacts are modified too, and will be downloaded together on the next synchronization.

## Downloads

- ◆ **download\_cursor** The **download\_cursor** script for Contact is as follows:

```

SELECT contact_id, contact.name, contact.cust_id
FROM ( contact JOIN customer ) JOIN salesrep
ON contact.cust_id = customer.cust_id
   AND customer.rep_id = salesrep.rep_id
WHERE Contact.last_modified >= ?
   AND salesrep.ml_username = ?
   AND Contact.active = 1

```

This script retrieves all contacts that are active, that have been changed since the last time the sales representative downloaded (either explicitly or by modification of the corresponding customer), and that are assigned to the representative. A join with the Customer and SalesRep table is needed to identify the contacts associated with this representative.

- ◆ **download\_delete\_cursor** The **download\_delete\_cursor** for Contact is as follows:

```

SELECT contact_id
FROM ( Contact JOIN Customer ) JOIN SalesRep
ON Contact.cust_id = Customer.cust_id
   AND Customer.rep_id = SalesRep.rep_id
WHERE Contact.last_modified >= ?
   AND Contact.active = 0

```

The automatic use of cascading referential integrity by the MobiLink client deletes contacts when the corresponding customer is deleted from the remote database. The **download\_delete\_cursor** script therefore has to delete only those contact explicitly marked as inactive.

## Uploads

Contact information can be inserted, updated, or deleted at the remote database. The scripts corresponding to these operations are as follows:

- ◆ **upload\_insert** The following **upload\_insert** script adds a row to the Contact table, marking the contact as active:

```
INSERT INTO Contact (
    contact_id, name, cust_id, active )
VALUES ( ?, ?, ?, 1 )
```

- ◆ **upload\_update** The following **upload\_update** script modifies the contact information at the consolidated database:

```
UPDATE Contact
SET name = ?, cust_id = ?
WHERE contact_id = ?
```

Conflict detection is not carried out on this table.

- ◆ **upload\_delete** The following **upload\_delete** script marks the contact as inactive at the consolidated database. It does not delete a row.

```
UPDATE Contact
SET active = 0
WHERE contact_id = ?
```

## Synchronizing products in the Contact sample

The scripts for the Product table illustrate conflict detection and resolution.

The Product table is kept in a separate publication from the other tables so that it can be downloaded separately. For example, if the price changes and the sales representative is synchronizing over a slow link, they can download the product changes without uploading their own customer and contact changes.

Business rules

The only change that can be made at the remote database is to change the quantity column, when an order is taken.

Downloads

- ◆ **download\_cursor** The following **download\_cursor** script downloads all rows changed since the last time the remote database synchronized:

```
SELECT id, name, size, quantity, unit_price
FROM product
WHERE last_modified >= ?
AND active = 1
```

- ◆ **download\_delete\_cursor** The following **download\_delete\_cursor** script removes all products no longer sold by the company. These products are marked as inactive in the consolidated database.

```
SELECT id, name, size, quantity, unit_price
FROM product
WHERE last_modified >= ?
AND active = 0
```

---

## Uploads

Only UPDATE operations are uploaded from the remote database. The major feature of these upload scripts is a conflict detection and resolution procedure.

If two sales representatives take orders and then synchronize, each order is subtracted from the quantity column of the Product table. For example, if Sam Singer takes an order for 20 baseball hats (product ID 400), he will change the quantity from 90 to 70. If Pam Savarino takes an order for 10 baseball hats before receiving this change, she will change the column in her database from 90 to 80.

When Sam Singer synchronizes his changes, the quantity column in the consolidated database is changed from 90 to 70. When Pam Savarino synchronizes her changes, the correct action is to set the value to 60. This setting is accomplished by detecting the conflict.

The conflict detection scheme includes the following scripts:

- ◆ **upload\_update** The following **upload\_update** script is a straightforward UPDATE at the consolidated database:

```
UPDATE product
SET name = ?, size = ?, quantity = ?, unit_price = ?
WHERE product.id = ?
```

- ◆ **upload\_fetch** The following **upload\_fetch** script fetches a single row from the Product table for comparison with the old values of the uploaded row. If the two rows differ, a conflict is detected.

```
SELECT id, name, size, quantity, unit_price
FROM Product
WHERE id = ?
```

- ◆ **upload\_old\_row\_insert** If a conflict is detected, the old values are placed into the product\_conflict table for use by the **resolve\_conflict** script. The row is added with a value of O (for Old) in the row\_type column.

```
INSERT INTO DBA.product_conflict(
    id, name, size, quantity, unit_price, row_type )
VALUES( ?, ?, ?, ?, ?, 'O' )
```

- ◆ **upload\_new\_row\_insert** The following script adds the new values of the uploaded row into the product\_conflict table for use by the **resolve\_conflict** script:

```
INSERT INTO DBA.product_conflict(
    id, name, size, quantity, unit_price, row_type )
VALUES( ?, ?, ?, ?, ?, 'N' )
```

Conflict resolution

- ◆ **resolve\_conflict** The following script resolves the conflict by adding the difference between new and old rows to the quantity value in the consolidated database:

```
UPDATE Product
SET p.quantity = p.quantity
                - old_row.quantity
                + new_row.quantity
FROM Product p,
     DBA.product_conflict old_row,
     DBA.product_conflict new_row
WHERE p.id = old_row.id
      AND p.id = new_row.id
      AND old_row.row_type = 'O'
      AND new_row.row_type = 'N'
```

---

## Monitoring statistics and errors in the Contact sample

The Contact sample contains some simple error reporting and monitoring scripts. The SQL statements to create these scripts are in the file *Samples\MobiLink>Contact\mlmaint.sql*.

The scripts insert rows into tables created to hold the values. For convenience, the tables are owned by a distinct user, mlmaint.

---

## CHAPTER 7

# The CustDB Sample Application

### About this chapter

This chapter uses the CustDB sample application to illustrate a variety of techniques that you can use for common synchronization tasks.

The techniques are illustrated using SQL scripts and Java synchronization logic. Many of the same techniques can be implemented using .NET synchronization logic.

### Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Introduction</a>	100
<a href="#">Setup</a>	102
<a href="#">Tables in the CustDB databases</a>	109
<a href="#">Users in the CustDB sample</a>	112
<a href="#">Synchronization</a>	113
<a href="#">Maintaining the customer and order primary key pools</a>	117
<a href="#">Further reading</a>	119

# Introduction

This chapter introduces you to the CustDB (Customer Database) MobiLink sample application. CustDB is a sales-status application.

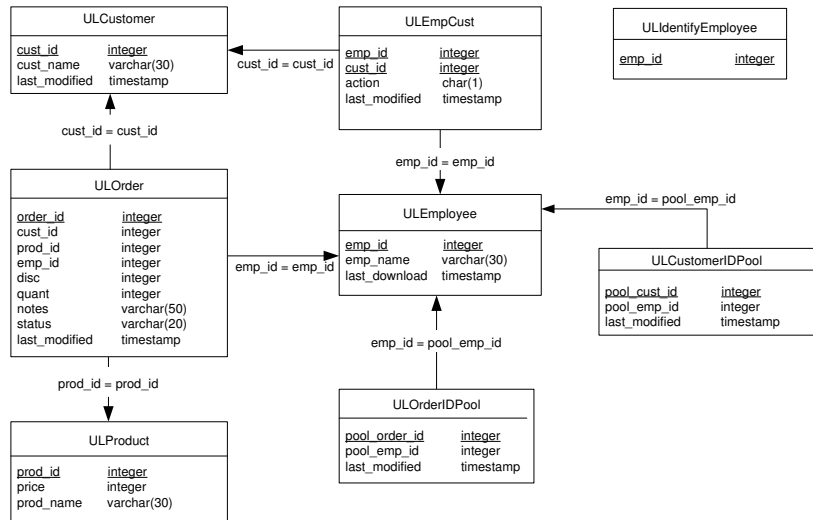
The CustDB sample is a valuable resource for the MobiLink developer. It provides you with examples of how to implement many of the techniques you will need to develop MobiLink applications.

The application has been designed to illustrate several common synchronization techniques. To get the most out of this chapter, you should study the sample application as you read.

A version of CustDB is supplied for each supported operating system and for each supported database type.

☞ For the locations of CustDB and setup instructions, see [“Setting up the CustDB consolidated database” on page 102.](#)

Following is the schema of CustDB:



## Scenario

The CustDB scenario is as follows.

A consolidated database is located at the head office. The following data is stored in the consolidated database:

- ◆ The MobiLink system tables that hold the synchronization metadata.
- ◆ The synchronization scripts that implement synchronization logic.
- ◆ The CustDB data, including all customer, product, and order information, stored in the rows of base tables.



There are two types of remote databases, mobile managers and sales representatives.

Each mobile sales representative's database contains all products but only those orders assigned to that sales representative, while a mobile manager's database contains all products and orders.

#### Synchronization design

The synchronization design in the CustDB sample application uses the following features:

- ◆ **Complete table downloads** All rows and columns of the ULProduct table are shared in their entirety with the remote databases.
- ◆ **Column subsets** All rows, but not all columns, of the ULCustomer table are shared with the remote databases.
- ◆ **Row subsets** Different remote users get different sets of rows from the ULOrder table.
  - ☞ For more information about row subsets, see [“Partitioning rows among remote databases”](#) [*MobiLink Administration Guide*, page 52].
- ◆ **Timestamp-based synchronization** This is a way of identifying changes that were made to the consolidated database since the last time a device synchronized. The ULCustomer and ULOrder tables are synchronized using a method based on timestamps.
  - ☞ For more information, see [“Timestamp-based synchronization”](#) [*MobiLink Administration Guide*, page 48].
- ◆ **Snapshot synchronization** This is a simple method of synchronization that downloads all rows in every synchronization. The ULProduct table is synchronized in this way.
  - ☞ For more information, see [“Snapshot synchronization”](#) [*MobiLink Administration Guide*, page 50].
- ◆ **Primary key pools to maintain unique primary keys** It is essential to ensure that primary key values are unique across a complete MobiLink installation. The primary key pool method used in this application is one way of ensuring unique primary keys.
  - ☞ For more information, see [“Maintaining unique primary keys using key pools”](#) [*MobiLink Administration Guide*, page 60].
  - ☞ For other ways to ensure that primary keys are unique, see [“Maintaining unique primary keys”](#) [*MobiLink Administration Guide*, page 56].

---

## Setup

This section describes the pieces that make up the code for the CustDB sample application and database. These include:

- ◆ The sample SQL scripts, located in the *Samples\MobiLink\CustDB* subdirectory of your SQL Anywhere installation.
- ◆ The application code, located in *Samples\UltraLite\CustDB*.
- ◆ Platform-specific user interface code, located in subdirectories of *Samples\UltraLite\CustDB* named for each operating system.

### Setting up the CustDB consolidated database

The consolidated database may be Adaptive Server Anywhere, Sybase Adaptive Server Enterprise, Microsoft SQL Server, Oracle, or IBM DB2.

The following SQL scripts are provided in the *Samples\MobiLink\CustDB* subdirectory of your SQL Anywhere 9 installation to build the consolidated database on any of these platforms:

- ◆ For an Adaptive Server Anywhere database, the file is *custdb.sql*.
- ◆ For an IBM DB2 database, the file is *custdb2.sql*.
- ◆ For an Adaptive Server Enterprise database, the file is *custase.sql*.
- ◆ For a Microsoft SQL Server database, the file is *custmss.sql*.
- ◆ For an Oracle database, the file is *custora.sql*.

### Creating a consolidated database

The following procedures create a consolidated database for CustDB for each of the supported types of consolidated database.

For databases other than Adaptive Server Anywhere databases, you will first need to run a script to add the MobiLink system tables. This platform-specific script is located in the *MobiLink\setup* subdirectory of your SQL Anywhere 9 installation.

☞ For more information about preparing a database for use as a consolidated database, see “[Setting up a consolidated database](#)” [*MobiLink Administration Guide*, page 33].

**❖ To set up a consolidated database (Adaptive Server Enterprise, Oracle or SQL Server)**

1. Create the consolidated database.
2. Add the MobiLink system tables by running one of the following SQL scripts, located in the *MobiLink\setup* subdirectory of your SQL Anywhere 9 installation:
  - ◆ For an Adaptive Server Enterprise consolidated database prior to version 12.5, run *syncase.sql*. Otherwise, run *syncase125.sql*.
  - ◆ For an Oracle consolidated database, run *syncora.sql*.
  - ◆ For a SQL Server consolidated database, run *syncmss.sql*.
3. Add tables to the CustDB database by running one of the following SQL scripts, located in the *Samples\MobiLink\CustDB* subdirectory of your SQL Anywhere 9 installation:
  - ◆ For an Adaptive Server Enterprise consolidated database, run *custase.sql*.
  - ◆ For an Oracle consolidated database, run *custora.sql*.
  - ◆ For a SQL Server consolidated database, run *custmss.sql*.
4. Create an ODBC data source called CustDB that references your database on the client machine.
  - ◆ Choose Start ► Programs ► Sybase SQL Anywhere 9 ► Adaptive Server Anywhere ► ODBC Administrator.
  - ◆ Click Add.
  - ◆ Select the appropriate driver from the list. Click Finish.
  - ◆ Name the ODBC data source CustDB.
  - ◆ Click the Login tab. Enter the user ID and password for your database.

**❖ To set up a consolidated database (Adaptive Server Anywhere)**

1. Create the consolidated database:

Navigate to the *Samples\MobiLink\CustDB* subdirectory of your SQL Anywhere 9 installation and run the following command line:

```
dbinit consol.db
```
2. Add tables to the CustDB database by running *custdb.sql*, located in the *Samples\MobiLink\CustDB* subdirectory of your SQL Anywhere 9 installation.

- 
- ◆ Choose Start ► Programs ► Sybase SQL Anywhere 9 ► Sybase Central.
  - ◆ In the right pane of Sybase Central, right-click Adaptive Server Anywhere 9 and connect to the consolidated database you have created. The default user ID and password are DBA and SQL.
  - ◆ In the right pane, right-click the consolidated database and select Open Interactive SQL from the popup menu.
  - ◆ In Interactive SQL, select File ► Run Script. Browse to *custdb.sql*. Click Open.
  - ◆ Close Interactive SQL.
3. Create an ODBC data source called CustDB that references your database on the client machine.
- ◆ In Sybase Central, select Tools ► Adaptive Server Anywhere 9 ► Open ODBC Administrator.
  - ◆ Click Add.
  - ◆ Select Adaptive Server Anywhere 9.0 Driver. Click Finish.
  - ◆ Name the ODBC data source CustDB.
  - ◆ Click the Login tab. Enter the user ID and password for your database. The default values are DBA and SQL.
  - ◆ Click the Database tab. Browse to the location of your database file.

#### ❖ To set up a consolidated database (IBM DB2)

1. Create a DB2 database on the DB2 server. Ensure that the default table space (usually called USERSPACE1) uses 8 Kb pages.  
If the default table space does not use 8 Kb pages, complete the following steps:
  - ◆ Verify that at least one of your buffer pools has 8 Kb pages. If not, create a buffer pool with 8 Kb pages.
  - ◆ Create a new table space and temporary table space with 8 Kb pages. For more information, consult your DB2 documentation.
2. Add the MobiLink system tables using the file *MobiLink\setup\syncdb2long.sql*.
  - ◆ Change the connect command in *syncdb2long.sql*. Replace *DB2Database* with the name of your ODBC data source. In this example, the ODBC data source is CustDB. You could also add the user name and password as follows. Replace *userid* and *password* with your user name and password.

```
connect to DB2Database user userid using password ~
```

- ◆ Open a DB2 Command Window on either the server or client computer. Run *syncdb2long.sql* by typing the following command:

```
db2 -c -ec -td~ +s -v -f syncdb2long.sql
```

3. In order for DB2 to use the stored procedures defined in *syncdb2long.sql*, you must copy the *syncdb2long\_version* Java and class files located in the *MobiLink\setup* subdirectory of your SQL Anywhere installation to the *FUNCTION* subdirectory of your DB2 installation.
4. Copy *custdb2.class* located in the *Samples\MobiLink\CustDB* subdirectory of your SQL Anywhere installation to the *SQLLIB\FUNCTION* directory on your DB2 server machine. This class contains procedures used for the CustDB sample.
5. Add tables to the CustDB database:

- ◆ If necessary, change the connect command in *custdb2.sql*. For example, you could add the user name and password as follows. Replace *userid* and *password* with your user name and password.

```
connect to CustDB user userid using password
```

- ◆ Open a DB2 Command Window on either the server or client computer. Run *custdb2.sql* by typing the following command:

```
db2 -c -ec -td~ +s -v -f custdb2.sql
```

- ◆ When processing is complete, enter the following command to close the command window:

```
exit
```

6. Create an ODBC data source called CustDB that references the DB2 database on the DB2 client machine.
  - ◆ Start the ODBC Administrator:
    - From the Start menu, choose Programs ► SQL Anywhere 9 ► Adaptive Server Anywhere ► ODBC Administrator.
    - The ODBC Data Source Administrator appears.
  - ◆ On the User DSN tab, click Add.
    - The Create New Data Source dialog appears.
  - ◆ Select iAnywhere Solutions 9 - DB2 Wire Protocol and click Finish.
    - The ODBC DB2 Wire Protocol Driver Setup dialog appears.
  - ◆ On the General tab, type the Data Source Name **db2\_consolidated** and enter the appropriate Ip Address, and Tcp port. For the IBM DB2 sample database use SAMPLE for the database name.

- 
- ◆ On the Bind tab, click Create Package. Enter the Username and Password for your IBM DB2 instance.
  - ◆ Click OK.
7. Run the *custdb2setuplong* Java application on the DB2 client machine as follows. This application resets the CustDB example in the DB2 database. After the initial setup, you can run this application at any time to reset the DB2 CustDB database by typing the same command line.
- ◆ If you use a name other than CustDB for the data source, you must modify the connection code in *custdb2setuplong.java* and recompile it as follows. If the path specified by the system variable *%db2tempdir%* contains spaces, you must enclose the path in quotation marks.

```
javac -g -classpath %db2tempdir%\java\jdk\lib\
      classes.zip;
      %db2tempdir%\java\db2java.zip;
      %db2tempdir%\java\runtime.zip custdb2setuplong.java
```

- ◆ Type the following, where *userid* and *password* are the user name and password for connecting to the CustDB ODBC data source.

```
java custdb2setuplong userid password
```

## Setting up an UltraLite remote database

The following procedure creates a remote database for CustDB. The CustDB remote database must be an UltraLite database.

The application logic for the remote database is located in the *Samples\UltraLite\CustDB* subdirectory of your SQL Anywhere 9 installation. It includes the following files:

- ◆ **Embedded SQL logic** The file *custdb.sqc* contains the SQL statements needed to query and modify information from the UltraLite database and the calls required to start synchronization with the consolidated database.
- ◆ **C++ API logic** The file *custdbapi.cpp* contains the C++ API logic.
- ◆ **User-interface features** These features are stored separately, in platform-specific subdirectories of *Samples\UltraLite\CustDB*.

You will complete the following steps in order to install the sample application to a remote device that is running UltraLite:

**❖ To install the sample application to a remote device**

1. Start the consolidated database.
2. Start the MobiLink synchronization server.
3. Install the sample application to your remote device.
4. Start the sample application on the remote device.
5. Synchronize the sample application.

**Example**

The following example installs the CustDB sample on a Palm device running against a DB2 consolidated database.

1. Ensure that the consolidated database is running:
  - ◆ For a DB2 database, open a DB2 Command Window and run the following command line, where *userid* and *password* are the user ID and password for connecting to the DB2 database:

```
db2 connect to CustDB user userid using password
```

2. Start the MobiLink synchronization server:
  - ◆ For a DB2 database, run the following command at a command prompt:

```
dbmlsrv9 -c "DSN=CustDB" -zp
```

3. Install the sample application to your Palm device:
  - ◆ On your PC, start Palm Desktop.
  - ◆ Click Quick Install on the Palm Desktop toolbar.
  - ◆ Click Add. Browse to *custdb.prc* in the *UltraLite\palm\68k* subdirectory of your SQL Anywhere 9 installation.
  - ◆ Click Open.
  - ◆ HotSync your Palm device.
4. Start the CustDB sample application on your Palm device:
  - ◆ Place your Palm device in its cradle.

When you start the sample application for the first time, you are prompted to synchronize to download an initial copy of the data. This step is required only the first time you start the application. After that, the downloaded data is stored in the UltraLite database.
  - ◆ Launch the sample application.

From the Applications view, tap CustDB.  
An initial dialog appears, prompting you for an employee ID.

- 
- ◆ Enter an employee ID.

For the purpose of this tutorial, enter a value of 50. The sample application also allows values of 51, 52, or 53, but behaves slightly differently in these cases.

☞ For more information about the behavior of each user ID, see [“Users in the CustDB sample” on page 112](#).

A message box tells you that you must synchronize before proceeding.

- ◆ Synchronize your application.

Use HotSync to obtain an initial copy of the data.

- ◆ Confirm that the data has been synchronized into the application.

From the Applications view, tap the CustDB application. The display shows an entry sheet for a customer, with entries.

5. Synchronize the remote application with the consolidated database. You will only need to complete this step when you have made changes to the database.

- ◆ Ensure that the consolidated database and the MobiLink synchronization server are running.

- ◆ Place the Palm device in its cradle.

- ◆ Press the HotSync button to synchronize.

## Clean up

You may want to reset the data in the CustDB database in order to restart the sample. To revert the data in the CustDB UltraLite database to its original state, complete the following steps.

### ❖ To reset the data in the sample application

1. Install the ULUtil on your device:

- ◆ For a Palm device, start Palm Desktop on your PC.
- ◆ Click Install on the Palm Desktop toolbar.
- ◆ Click Add. Browse to *ulutil.prc* in the *UltraLite\palm\68k* subdirectory of your SQL Anywhere 9 installation.
- ◆ Click Done.
- ◆ HotSync your Palm device.

2. Delete the data using ULUtil:

- ◆ For a Palm device, tap the ULUtil icon.
- ◆ Select CustDB and tap Delete Data.
- ◆ HotSync your Palm device.



## Tables in the CustDB databases

The table definitions for the CustDB database are in platform-specific files in the *Samples\MobiLink\CustDB* subdirectory of your SQL Anywhere 9 installation.

Both the consolidated and the remote databases contain the following five tables, although their definitions are slightly different in each location.

### ULCustomer

The ULCustomer table contains a list of customers.

In the remote database, ULCustomer has the following columns:

- ◆ **cust\_id** A primary key column that holds a unique integer identifying the customer.
- ◆ **cust\_name** A 30-character string containing the name of the customer.

In the consolidated database, ULCustomer has the following additional column:

- ◆ **last\_modified** A timestamp containing the last time the row was modified. This column is used for timestamp-based synchronization.

### ULProduct

The ULProduct table contains a list of products.

In the both the remote and consolidated databases, ULProduct has the following columns:

- ◆ **prod\_id** A primary key column that holds a unique integer identifying the product.
- ◆ **price** An integer identifying the unit price.
- ◆ **prod\_name** A 30-character string containing the name of the product.

### ULOrder

The ULOrder table contains a list of orders, including details of the customer who placed the order, the employee who took the order, and the product being ordered.

In the remote database, ULOrder has the following columns:

- ◆ **order\_id** A primary key column that holds a unique integer identifying the order.
- ◆ **cust\_id** A foreign key column referencing ULCustomer.
- ◆ **prod\_id** A foreign key column referencing ULProduct.
- ◆ **emp\_id** A foreign key column referencing ULEmployee.
- ◆ **disc** An integer containing the discount applied to the order.

- 
- ◆ **quant** An integer containing the number of products ordered.
  - ◆ **notes** A 50-character string containing notes about the order.
  - ◆ **status** A 20-character string describing the status of the order.

In the consolidated database, ULOrder has the following additional column:

- ◆ **last\_modified** A timestamp containing the last time the row was modified. This column is used for timestamp-based synchronization.

#### ULOrderIDPool

The ULOrderIDPool table is a primary key pool for ULOrder.

In the remote database, ULOrderIDPool has the following column:

- ◆ **pool\_order\_id** A primary key column that holds a unique integer identifying the order ID.

In the consolidated database, ULOrderIDPool has the following additional columns:

- ◆ **pool\_emp\_id** An integer column containing the employee ID of the owner of the remote database to which the order ID has been assigned.
- ◆ **last\_modified** A timestamp containing the last time the row was modified.

#### ULCustomerIDPool

The ULCustomerIDPool table is a primary key pool for ULCustomer.

In the remote database, ULCustomerIDPool has the following column:

- ◆ **pool\_cust\_id** A primary key column that holds a unique integer identifying the customer ID.

In the consolidated database, ULCustomerIDPool has the following additional columns:

- ◆ **pool\_emp\_id** An integer column containing the employee ID that will be used for a new employee generated at a remote database.
- ◆ **last\_modified** A timestamp containing the last time the row was modified.

The following tables are contained in the consolidated database only:

#### ULIdentifyEmployee\_ nosync

The ULIdentifyEmployee\_nosync table exists only in the consolidated database. It has a single column as follows:

- ◆ **emp\_id** This primary key column contains an integer representing an employee ID.

ULEmployee	<p>The ULEmployee table exists only in the consolidated database. It contains a list of sales employees.</p> <p>ULEmployee has the following columns:</p> <ul style="list-style-type: none"> <li>◆ <b>emp_id</b> A primary key column that holds a unique integer identifying the employee.</li> <li>◆ <b>emp_name</b> A 30-character string containing the name of the employee.</li> </ul>
ULEmpCust	<p>The ULEmpCust table controls which customers' orders will be downloaded. If the employee needs a new customer's orders, inserting the employee ID and customer ID will force the orders for that customer to be downloaded.</p> <ul style="list-style-type: none"> <li>◆ <b>emp_id</b> A foreign key to ULEmployee.emp_id.</li> <li>◆ <b>cust_id</b> A foreign key to ULCustomer.cust_id. The primary key consists of emp_id and cust_id.</li> <li>◆ <b>action</b> A character used to determine if an employee record should be deleted from the remote database. If the employee no longer requires a customer's orders, set to D (delete). If the orders are still required, the action should be set to NULL.</li> </ul> <p>A logical delete must be used in this case so that the consolidated database can identify which rows to remove from the ULOrder table. Once the deletes have been downloaded, all records for that employee with an action of D can also be removed from the consolidated database.</p> <ul style="list-style-type: none"> <li>◆ <b>last_modified</b> A timestamp containing the last time the row was modified. This column is used for timestamp-based synchronization.</li> </ul>
ULOldOrder and ULNewOrder	<p>These tables exist only in the consolidated database. They are for conflict resolution and contain the same columns as ULOrder. In Adaptive Server Anywhere and Microsoft SQL Server these are temporary tables. In Adaptive Server Enterprise, these are normal tables and @@spid. DB2 and Oracle do not have temporary tables, so MobiLink needs to be able to identify which rows belong to the synchronizing user. Since these are base tables, if five users are synchronizing, they might each have a row in these tables at the same time.</p> <p>☞ For more information about @@spid, see <a href="#">“Variables” [ASA SQL Reference, page 38]</a>.</p>

---

## Users in the CustDB sample

There are two types of users in the CustDB sample, sales people and mobile managers. The differences are as follows:

- ◆ **Sales people** User IDs 51, 52, and 53 identify remote databases that are associated with sales people. Sales people can carry out the following tasks:
  - View lists of customers and products.
  - Add new customers.
  - Add or delete orders.
  - Scroll through the list of outstanding orders.
  - Accept or deny orders.
  - Synchronize changes with the consolidated database.
- ◆ **Mobile managers** User ID 50 identifies the remote database associated with the mobile manager. The mobile manager can perform the same tasks as a sales person. In addition, the mobile manager can do the following:
  - Accept or deny orders.

## Synchronization

The following sections describe the CustDB sample's synchronization logic.

### Synchronization logic source code

You can use Sybase Central to inspect the synchronization scripts in the consolidated database.

Script types and events	The <i>custdb.sql</i> file adds each synchronization script to the consolidated database by calling <code>ml_add_connection_script</code> or <code>ml_add_table_script</code> .
Example	The following lines in <i>custdb.sql</i> add a table-level script for the <code>ULProduct</code> table, which is executed during the <code>download_cursor</code> event. The script consists of a single <code>SELECT</code> statement.

```
call ml_add_table_script(
  'CustDB',
  'ULProduct', 'download_cursor',
  'SELECT prod_id, price, prod_name FROM ULProduct' )
go
```

### Synchronizing orders in the CustDB sample

Business rules	<p>The business rules for the <code>ULOrder</code> table are as follows:</p> <ul style="list-style-type: none"> <li>◆ Orders are downloaded only if they are not approved or the status is null.</li> <li>◆ Orders can be modified at both the consolidated and remote databases.</li> <li>◆ Each remote database contains only the orders assigned to an employee.</li> </ul>
Downloads	<p>Orders can be inserted, deleted or updated at the consolidated database. The scripts corresponding to these operations are as follows:</p> <ul style="list-style-type: none"> <li>◆ <b>download_cursor</b> The first parameter in the <b>download_cursor</b> script is the last download timestamp. It is used to ensure that only rows that have been modified on either the remote or the consolidated database since the last synchronization are downloaded. The second parameter is the employee ID. It is used to determine which rows to download.</li> </ul>

The **download\_cursor** script for CustDB is as follows:

```
CALL ULOrderDownload( ?, ? )
```

The **ULOrderDownload** procedure for CustDB is as follows:

---

```

ALTER PROCEDURE UOrderDownload ( IN LastDownload timestamp,
                                IN EmployeeID integer )
BEGIN
    SELECT o.order_id, o.cust_id, o.prod_id, o.emp_id, o.disc,
           o.quant, o.notes, o.status
    FROM UOrder o, UEmpCust ec
    WHERE o.cust_id = ec.cust_id
    AND ec.emp_id = EmployeeID
    AND ( o.last_modified >= LastDownload
    OR ec.last_modified >= LastDownload)
    AND ( o.status IS NULL OR o.status != 'Approved' )
    AND ( ec.action IS NULL )
END

```

- ◆ **download\_delete\_cursor** The **download\_delete\_cursor** script for CustDB is as follows:

```

SELECT o.order_id, o.cust_id, o.prod_id, o.emp_id, o.disc,
       o.quant, o.notes, o.status
FROM UOrder o, UEmpCust ec
WHERE o.cust_id = ec.cust_id
AND ( ( o.status = 'Approved' AND o.last_modified >= ? )
OR ( ec.action = 'D' ) )
AND ec.emp_id = ?

```

## Uploads

Orders can be inserted, deleted or updated at the remote database. The scripts corresponding to these operations are as follows:

- ◆ **upload\_insert** The **upload\_insert** script for CustDB is as follows:

```

INSERT INTO "UOrder" ( "order_id", "cust_id", "prod_id",
                      "disc", "quant", "notes", "status" )
VALUES ( ?, ?, ?, ?, ?, ?, ? )

```

- ◆ **upload\_update** The **upload\_update** script for CustDB is as follows:

```

UPDATE UOrder SET cust_id=?, prod_id=?, emp_id=?, disc=?,
                 quant=?, notes=?, status=?
WHERE order_id = ?

```

- ◆ **upload\_delete** The **upload\_delete** script for CustDB is as follows:

```

DELETE FROM "UOrder" WHERE "order_id" = ?

```

- ◆ **upload\_fetch** The **upload\_fetch** script for CustDB is as follows:

```

SELECT order_id, cust_id, prod_id, emp_id, disc, quant, notes,
       status
FROM UOrder WHERE order_id = ?

```

- ◆ **upload\_old\_row\_insert** The **upload\_old\_row\_insert** script for CustDB is as follows:

```
INSERT INTO ULOldOrder ( order_id, cust_id, prod_id, emp_id,
                        disc, quant, notes, status )
VALUES( ?, ?, ?, ?, ?, ?, ?, ? )
```

- ◆ **upload\_new\_row\_insert** The **upload\_new\_row\_insert** script for CustDB is as follows:

```
INSERT INTO ULNewOrder ( order_id, cust_id, prod_id, emp_id,
                        disc, quant, notes, status )
VALUES( ?, ?, ?, ?, ?, ?, ?, ? )
```

Conflict resolution

- ◆ **resolve\_conflict** The **resolve\_conflict** script for CustDB is as follows:

```
CALL ULResolveOrderConflict
```

The **ULResolveOrderConflict** procedure for CustDB is as follows:

```
ALTER PROCEDURE ULResolveOrderConflict()
BEGIN
-- approval overrides denial
IF 'Approved' = (SELECT status FROM ULNewOrder) THEN
    UPDATE ULOrder o
    SET o.status = n.status, o.notes = n.notes
    FROM ULNewOrder n
    WHERE o.order_id = n.order_id;
END IF;
DELETE FROM ULOldOrder;
DELETE FROM ULNewOrder;
END
```

## Synchronizing customers in the CustDB sample

Business rules

The business rules governing customers are as follows:

- ◆ Customer information can be modified at both the consolidated and remote databases.
- ◆ Both the remote and consolidated databases contain a complete listing of customers.

Downloads

Customer information can be inserted or updated at the consolidated database. The script corresponding to these operations is as follows:

- ◆ **download\_cursor** The following **download\_cursor** script downloads all customers for whom information has changed since the last time the user downloaded information.

```
SELECT cust_id, cust_name FROM ULCustomer WHERE last_modified >=
?
```

Uploads

Customer information can be inserted, updated, or deleted at the remote database. The scripts corresponding to these operations are as follows:

- 
- ◆ **upload\_insert** The **upload\_insert** script for CustDB is as follows:

```
INSERT INTO ULCustomer ( cust_id, cust_name ) VALUES ( ?, ?
)
```

- ◆ **upload\_update** The **upload\_update** script for CustDB is as follows:

```
UPDATE ULCustomer SET cust_name = ?
WHERE "cust_id" = ?
```

Conflict detection is not carried out on this table.

- ◆ **upload\_delete** The **upload\_delete** script for CustDB is as follows:

```
DELETE FROM ULCustomer WHERE cust_id = ?
```

## Synchronizing products in the CustDB sample

### Business rules

The business rules for the ULProduct table are as follows:

- ◆ Products can only be modified at the consolidated database.
- ◆ Each remote database contains all of the products.

### Downloads

Product information can be inserted, deleted, or updated at the consolidated database. The script corresponding to these operations is as follows:

- ◆ **download\_cursor** The following **download\_cursor** script downloads all of the rows and columns of the ULProduct table at each synchronization:

```
SELECT prod_id, price, prod_name FROM ULProduct
```



## Maintaining the customer and order primary key pools

The CustDB sample database uses primary key pools in order to maintain unique primary keys in the ULCustomer and ULOrder tables. The primary key pools are the ULCustomerIDPool and ULOrderIDPool tables.

### ULCustomerIDPool

The following scripts are defined in the ULCustomerIDPool table:

#### Downloads

- ◆ **download\_cursor** The **download\_cursor** script for CustDB is as follows:

```
SELECT pool_cust_id FROM ULCustomerIDPool
WHERE last_modified >= ?
AND pool_emp_id = ?
```

#### Uploads

- ◆ **upload\_insert** The **upload\_insert** script for CustDB is as follows:

```
INSERT INTO ULCustomerIDPool ( pool_cust_id ) VALUES ( ? )
```

- ◆ **upload\_delete** The **upload\_delete** script for CustDB is as follows:

```
DELETE FROM ULCustomerIDPool WHERE pool_cust_id = ?
```

- ◆ **end\_upload** This **end\_upload** script ensures that after each upload 20 customer IDs remain in the customer ID pool:

```
CALL ULCustomerIDPool_maintain( ? )
```

The **UL\_CustomerIDPool\_maintain** procedure for CustDB is as follows:

```
ALTER PROCEDURE ULCustomerIDPool_maintain ( IN syncuser_id
      INTEGER )
BEGIN
  DECLARE pool_count INTEGER;
  -- Determine how many ids to add to the pool
  SELECT COUNT(*) INTO pool_count
  FROM ULCustomerIDPool
  WHERE pool_emp_id = syncuser_id;
  -- Top up the pool with new ids
  WHILE pool_count < 20 LOOP
    INSERT INTO ULCustomerIDPool ( pool_emp_id )
      VALUES ( syncuser_id );
    SET pool_count = pool_count + 1;
  END LOOP;
END
```

---

## ULOrderIDPool

The following scripts are defined in the ULOrderIDPool table:

### Downloads

- ◆ **download\_cursor** The **download\_cursor** script for CustDB is as follows:

```
SELECT pool_order_id FROM ULOrderIDPool
WHERE last_modified >= ?
AND pool_emp_id = ?
```

### Uploads

- ◆ **end\_upload** This **end\_upload** script ensures that after each upload 20 order IDs remain in the order ID pool.

```
CALL ULOrderIDPool_maintain( ? )
```

The **UL\_OrderIDPool\_maintain** procedure for CustDB is as follows:

```
ALTER PROCEDURE ULOrderIDPool_maintain ( IN syncuser_id
INTEGER )
BEGIN
DECLARE pool_count INTEGER;
-- Determine how many ids to add to the pool
SELECT COUNT(*) INTO pool_count
FROM ULOrderIDPool
WHERE pool_emp_id = syncuser_id;
-- Top up the pool with new ids
WHILE pool_count < 20 LOOP
INSERT INTO ULOrderIDPool ( pool_emp_id )
VALUES ( syncuser_id );
SET pool_count = pool_count + 1;
END LOOP;
END
```

- ◆ **upload\_insert** The **upload\_insert** script for CustDB is as follows:

```
INSERT INTO ULOrderIDPool ( pool_order_id ) VALUES( ? )
```

- ◆ **upload\_delete** The **upload\_delete** script for CustDB is as follows:

```
DELETE FROM ULOrderIDPool WHERE pool_order_id = ?
```

## Further reading

The following documentation sections are good starting points for further reading:

☞ For more information about script types, see [“Script types”](#) [*MobiLink Administration Guide*, page 236].

☞ For reference material, including detailed information about each script and its parameters, see [“Synchronization Events”](#) [*MobiLink Administration Guide*, page 319].



# Index

## A

add version wizard  
 using 22

## C

client event-hook procedures *see also* event hooks  
 conflict resolution  
 Contact sample 95  
 CustDB sample 116  
 Contact MobiLink sample  
 about 84  
 building 85  
 Contact table 93  
 Customer table 91  
 monitoring statistics 98  
 Product table 95  
 running 85  
 SalesRep table 91  
 tables 87  
 users 90  
 conventions  
 documentation viii  
 create database wizard  
 using 15  
 custase.sql  
 location 102  
 CustDB application  
 DB2 102  
 MobiLink sample application 99  
 synchronization scripts 102  
 CustDB MobiLink sample  
 tables 109  
 ULCustomer table 115  
 ULOrder table 113  
 ULProduct table 116  
 users 112  
 custdb.sq  
 location 106  
 custdb.sql  
 location 102  
 custmss.sql

location 102  
 custora.sql  
 location 102

## D

DB2  
 CustDB tutorial 102  
 documentation  
 conventions viii  
 SQL Anywhere Studio vi

## F

feedback  
 documentation xi  
 providing xi

## H

hooks *see also* event hooks

## I

IBM DB2  
 CustDB tutorial 102  
 icons  
 used in manuals x

## J

Java  
 synchronization scripts tutorial 51

## L

log files  
 MobiLink 26

## M

MobiLink  
 Java tutorial 51  
 MobiLink Custdb tutorial 99  
 .NET tutorial 65  
 Oracle tutorial 39

Sybase Central tutorial	13	.NET tutorial	65
Tutorial - MobiLink sample applications	83	synchronization subscriptions <i>see also</i> subscriptions	
Using ASA tutorial	1	synchronization techniques	
MobiLink synchronization		custdb sample application	99
custdb sample database	99	MobiLink Contact sample tutorial	83
Java tutorial	51	syncora.sql	
.NET tutorial	65	using	43
MobiLink synchronization client tutorial	8		
MobiLink synchronization logic		<b>T</b>	
Java tutorial	51	technical support	
.NET tutorial	65	newsgroups	xi
MobiLink synchronization server tutorial	6	tutorials	
		MobiLink Contact sample	83
<b>N</b>		MobiLink custdb sample	99
.NET		MobiLink Java logic	51
MobiLink tutorial	65	MobiLink .NET logic	65
newsgroups		MobiLink with ASA clients	1
technical support	xi	MobiLink with Oracle	39
		Monitoring MobiLink scripts and conflict resolution	13
<b>O</b>			
Oracle		<b>U</b>	
MobiLink tutorial	39	upload_delete	
		Contact sample	95
<b>S</b>		CustDB sample	116
sample application			
MobiLink CustDB application	99	<b>W</b>	
sample database		wizards	
MobiLink CustDB application	99	add version	22
samples		writing synchronization scripts in Java tutorial	51
Contact MobiLink sample	84	writing synchronization scripts in .NET tutorial	65
MobiLink CustDB application	99		
schemas			
CustDB sample database	100		
SQL Anywhere Studio			
documentation	vi		
support			
newsgroups	xi		
synchronization			
Java tutorial	51		
MobiLink Oracle tutorial	39		
MobiLink Sybase Central tutorial	13		
MobiLink tutorial	1		
synchronization scripts			
Java tutorial	51		