UltraLite™ Component Suite Foundations

Last modified: October 2002
Part Number: 37121-01-0802-01

Last modified October 2002. Part number 37121-01-0802-01.

Contents

About This Manual

Subject	This manual introduces the UltraLite Component Suite. It is a companion to the *User's Guide* for your particular development platform.
Audience	This manual is intended for all UltraLite Component Suite developers who wish to take advantage of the performance, resource efficiency, robustness, and security of an UltraLite relational database for data storage and synchronization.

The UltraLite sample database

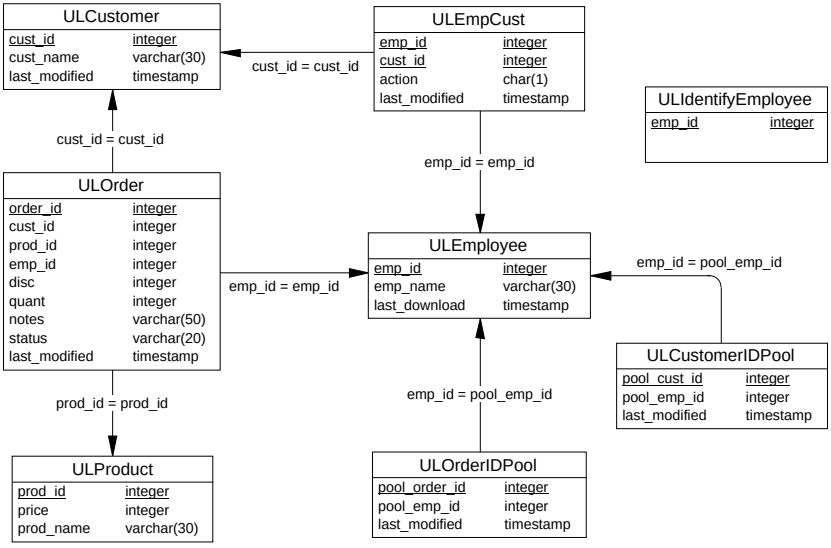
Some of the examples in the MobiLink and UltraLite documentation use the UltraLite sample database.

The UltraLite sample database is held in a file named *custdb.db*, or *custdb.xml*, and is located in the *Samples\UltraLite\CustDB* subdirectory of your SQL Anywhere directory. Complete applications built on this database are also supplied in the following directories:

Component	Application sample
UltraLite for MobileVB	*Samples\UltraLiteForAppForge\CustDB*
UltraLite for eMbedded Visual Basic	*Samples\UltraLiteActiveX\CustDB*
Native UltraLite for Java	*Samples\NativeUltraLiteForJava\CustDB*

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

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



Finding out more and providing feedback

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

You can provide feedback on this documentation and on the software through a newsgroup. The newsgroup can be found on the *forums.sybase.com* news server as news://forums.sybase.com/sybase.public.sqlanywhere.ultralite.

Newsgroup disclaimer

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

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

CHAPTER 1

Introduction to the UltraLite Component Suite

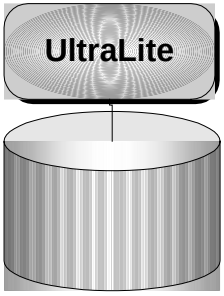
About this chapter

This chapter introduces you to UltraLite Component Suite features, platforms, architecture, and functionality.

Contents

Introduction to the UltraLite Component Suite

UltraLite is a relational database and synchronization technology for small, mobile, and embedded devices. UltraLite has been available for C/C++ developers and Java developers as part of SQL Anywhere Studio. The UltraLite Component Suite brings this proven technology to users of rapid application development systems.



UltraLite Component Suite benefits

The UltraLite Component Suite provides the following benefits to application developers:

- **Robust data management** Data held on small devices is as important as data in enterprise databases. UltraLite brings transaction processing, referential integrity, and other benefits of relational databases to small devices.

  For more information about UltraLite database features, see "UltraLite databases" on page 3.

- **Powerful synchronization** An information system is only as robust as its weakest link. UltraLite gives you the ability to synchronize data with a central database-management system when used with SQL Anywhere Studio.

  The UltraLite Component Suite uses MobiLink synchronization technology, included in SQL Anywhere Studio, to synchronize with industry-standard database-management systems. MobiLink synchronization works with ODBC-compliant data sources such as Sybase Adaptive Server Anywhere, Sybase Adaptive Server Enterprise, IBM DB2, Microsoft SQL Server, and Oracle. It provides flexible, programmable, and scalable synchronization that can manage thousands of UltraLite databases.

- **Straightforward development** An object-based programming interface provides straightforward access to data. Integration into popular development tools such as eMbedded Visual Basic, AppForge MobileVB, and Borland JBuilder makes developers productive. A graphical tool enables you to design and modify schemas for UltraLite databases rapidly.
- **Multi-platform availability** You can develop and deploy UltraLite database applications for Windows CE, Palm OS, and Java-based devices.

UltraLite databases

UltraLite databases are transaction-processing relational databases, and provide you with the following features:

- **Tables** A single UltraLite database can hold many tables. The number and type of columns in a relational database table is fixed at design time, but each table can have any number of rows (up to 64K). Each row has a single entry for each column. The special NULL entry is used when there is no value for the entry.

  When designing your database, each table should represent a separate type of item, such as Customers, Employees, and so on.
- **Indexes** The rows in a relational database table are not ordered. You can create indexes to access the rows in order. Indexes are commonly associated with a single column, but UltraLite also provides multi-column indexes.
- **Keys** Each table has a special index called the **primary key**. Entries in the primary key column or columns must be unique.

  Foreign keys relate the data in one table to that in another. Each entry in the foreign key column must correspond to an entry in the primary key of another table.

  Between them, primary keys and foreign keys ensure that the database has **referential integrity**. Referential integrity is enforced in UltraLite databases, so that you cannot (for example) enter an order for a customer unless that customer exists in the database.

  By enforcing referential integrity, UltraLite ensures that the data in your UltraLite database is correct, in the same manner that data elsewhere in the enterprise is correct.

- **Publications** If you wish to synchronize the data in your UltraLite database with other databases you must have a valid SQL Anywhere Studio license. SQL Anywhere Studio includes MobiLink synchronization technology to synchronize UltraLite databases with desktop, workgroup or enterprise databases.

  Publications define sets of data to be synchronized. It is often desirable to synchronize all the data in an UltraLite database, but publications provide extra flexibility and control. They allow you to perform priority synchronizations, which means you can specify that only certain tables or groups of tables should be synchronized.

- **Transactions and recovery** UltraLite has commit and rollback features, together with automatic recovery in the event of device failure, to guarantee that transactions are executed completely or not at all.

- **Data types** UltraLite databases can manage a full range of data types, as well as default values and NULL values.

- **Security** UltraLite provides user authentication and database encryption, as well as encryption on the device and during synchronization, to build secure applications.

- **Performance and small footprint** UltraLite target devices tend to have relatively slow processors. UltraLite employs algorithms and data structures that provide high performance and low memory use. For example, UltraLite provides a caching algorithm designed specifically for small devices.

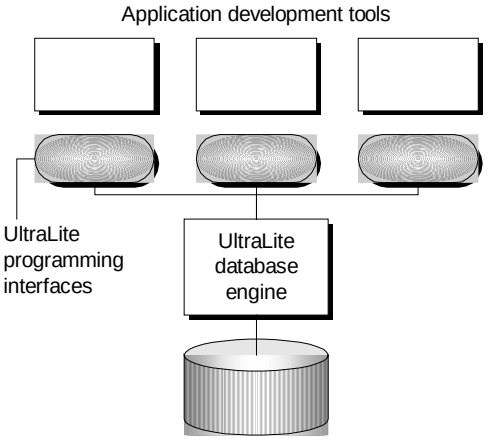
Rapid application development

The UltraLite Component Suite extends UltraLite to application development tools such as eMbedded Visual Basic, Java and AppForge MobileVB. An UltraLite component is provided for each development tool. Each component exposes a set of objects for data manipulation in the form of an API tailored to suit the expectations of users of a particular tool or language.

For an introduction to each component in the UltraLite Component Suite, see the following.

Application language	See
Java	"Introduction to Native UltraLite for Java" on page 1 of the book *UltraLite for Java User's Guide*
eMbedded Visual Basic	"Introduction to UltraLite for eMbedded Visual Basic" on page 1 of the book *UltraLite for eMbedded Visual Basic User's Guide*
AppForge MobileVB	"Introduction to UltraLite for MobileVB" on page 1 of the book *UltraLite for AppForge User's Guide*

All UltraLite components are built on the same underlying runtime.



Each component consists of a programming interface that exposes the UltraLite database functionality.

Applications that you create with an UltraLite Component will consist of the following:

- Your application code
- The UltraLite component
- An UltraLite database

System requirements and supported platforms

System requirements and supported platforms are discussed in this section.

Supported platforms

Platform support for UltraLite is of the following kinds:

- **Target platforms** The **target platform** is the device and operating system on which you deploy your UltraLite application.
- **Development platforms** For each target platform, you develop your applications using a particular development tool and operating system. The tool and operating system comprise the **development platform**.

Target platforms

The supported target platforms are given in the table below.

UltraLite Component	Target platforms
Native UltraLite for Java	Jeode Personal Java 1.2 compatible VM on Windows CE devices using the ARM processor, including the Compaq iPaq and NEC MobilePro P300.
	Windows operating systems other than Windows CE are supported for testing and development purposes only.
UltraLite for MobileVB	Windows CE 3.0 and higher, with Pocket PC on the ARM or MIPS processor.
	Palm OS version 3.1 and higher.
UltraLite for eMbedded Visual Basic	Windows CE 3.0 and higher, with Pocket PC on Arm or MIPS processors.
	The Windows CE emulator is also supported.

Supported development platform

To develop applications using UltraLite, you require application software as given in the table below.

UltraLite Component	Application development software
Native UltraLite for Java	JDK 1.1.8 for development.
	Jeode Personal Java 1.2 for deployment to Windows CE devices.
	Borland JBuilder 7 integration is provided.
UltraLite for MobileVB	AppForge MobileVB 3
	Microsoft Visual Basic 6.0
UltraLite for eMbedded Visual Basic	Microsoft eMbedded Visual Basic 3.0

SQL Anywhere Studio

The UltraLite Component Suite is available separately and as part of SQL Anywhere Studio. You require SQL Anywhere Studio to add the following capabilities to your UltraLite applications:

- **Synchronization** SQL Anywhere Studio users can synchronize the data in UltraLite applications with any ODBC-compliant central database.

  For more information, see "MobiLink synchronization features" on page 10.

- **Reference database** SQL Anywhere Studio users can generate an UltraLite schema file from an Adaptive Server Anywhere database.

  For more information, see "UltraLite initialization utility" on page 17.

The UltraLite Component Suite application development process

To develop an application using the UltraLite Component Suite you follow the following basic sequence of steps.

1 Design your database.

A database **schema** is the database definition, including all tables, indexes, and so on. You create a database schema using the UltraLite Schema Painter or writing an XML file. Users of SQL Anywhere Studio can generate an UltraLite database schema from an Adaptive Server Anywhere database.

UltraLite holds the database schema in a schema file. The UltraLite components use the information in this file to create a database when an application is first run.

For more information on the UltraLite Schema Painter, see "The UltraLite Schema Painter" on page 19.

For more information on the UltraLite utility *ulinit*, see "UltraLite initialization utility" on page 17.

2 Set up your development environment.

In each component, you are required to develop your application on a specific development platform, and to deploy to a specific target device. To achieve this end, you need to set up your development environment in tandem with the target environment. Tutorials in the companion books show you how you can accomplish this setup.

For more information on creating a project architecture for UltraLite for eMbedded Visual Basic, see "Adding the UltraLite component to the design environment" on page 28 of the book *UltraLite for eMbedded Visual Basic User's Guide*.

For more information on creating a project architecture for UltraLite for MobileVB, see the tutorial "Lesson 2: Create a project architecture" on page 11 of the book *UltraLite for AppForge User's Guide*.

For more information on creating a project architecture for Native UltraLite for Java, see "Understanding UltraLite Development" on page 33 of the book *UltraLite for Java User's Guide*.

3 Write your application code.

Create forms for your application and write code that includes:

- Code to create, open and connect to a database
- Code to access database tables using table objects
- Code to make use of insert, update and delete operations, and can take advantage of navigation procedures
- Code for synchronization, if required for your application.

4 Deploy your application to the device.

You can run the application in the development environment to confirm functionality, and you can configure application settings or synchronize your data to an enterprise database.

Each of these processes is outlined in detail for each application in each component book.

Synchronizing UltraLite applications

Users of SQL Anywhere Studio can synchronize UltraLite applications with a central database. This database may be a desktop database for personal applications, or a multi-user database for shared data, including enterprise data. Synchronization requires the MobiLink synchronization software included with SQL Anywhere Studio.

Synchronization details can be found in the *MobiLink Synchronization User's Guide* and the *UltraLite User's Guide* included with SQL Anywhere Studio documentation. This section provides a brief introduction to synchronization and describes some features of particular interest to users of the UltraLite Component Suite.

You can also find a working example of synchronization in the CustDB sample application.

MobiLink synchronization features

Many mobile and embedded computing applications are integrated into an information infrastructure. They require data to be uploaded to and downloaded from a **consolidated database**. This bi-directional sharing of information is **synchronization**.

MobiLink synchronization technology, included in SQL Anywhere Studio along with UltraLite, is designed to work with industry standard SQL database-management systems from Sybase and other vendors. UltraLite automatically keeps track of changes made to the UltraLite database between each synchronization with the consolidated database. When the UltraLite database is synchronized, all changes since the previous synchronization are uploaded.

Subset of the central database

Mobile and embedded databases need not contain all the data that exists in the consolidated database.

The tables in each UltraLite database can have a subset of the rows and columns in the central database. For example, a customer table might contain over 100 columns and 100 000 rows in the consolidated database, but the UltraLite database may only require 4 columns and 1000 rows. MobiLink allows you to define the exact subset to be downloaded to each remote database.

Flexibility

MobiLink synchronization is flexible. You define the subset of data using the native SQL dialect of the consolidated database-management system. Tables in the UltraLite database can correspond to tables in the consolidated database, but you can also populate an UltraLite table from a consolidated table with a different name, or from a join of one or more tables.

Conflict resolution

Mobile and embedded databases frequently share common data. They also must allow updates to the same data. When two or more remote databases simultaneously update the same row, the conflict cannot be prevented. It must be detected and resolved when the changes are uploaded to the central database. MobiLink synchronization automatically detects these conflicts. The conflict resolution logic is defined in the native SQL dialect of the central database.

The MobiLink synchronization server

An UltraLite application synchronizes with a central, consolidated database through the **MobiLink synchronization server**. This server provides an interface between the UltraLite application and the database server.

You control the synchronization process using **synchronization scripts**. These scripts may be SQL statements or procedures written in the native language of the consolidated DBMS, or they may be Java classes. For example, you can use a SELECT statement to identify the columns and tables in the consolidated database that correspond to each column of a row to be downloaded to a table in your UltraLite application. Each script controls a particular event during the synchronization process.

Adding synchronization to your application

This section provides a brief introduction to how to add synchronization to your application.

❖ **To control synchronization your application must carry out the following sequence of operations:**

1 Prepare the synchronization information.

Assign values to properties of the **ULSyncParms** object. In Native UltraLite for Java, use the Connection.SyncParms object.

2 Synchronize.

Call the ULConnection.Synchronize method. In Native UltraLite for Java, use the Connection.synchronize method.

For information about the properties and the values that you should set, look up **synchronization parameters: about** in the SQL Anywhere Studio online books index. For information on the synchronization methods, see the following:

- **MobileVB** See "ULSyncParms class" on page 93 of the book *UltraLite for AppForge User's Guide* and "Synchronize method" on page 72 of the book *UltraLite for AppForge User's Guide*.
- **eMbedded Visual Basic** See "ULSyncParms class" on page 84 of the book *UltraLite for eMbedded Visual Basic User's Guide* and "Synchronize method" on page 63 of the book *UltraLite for eMbedded Visual Basic User's Guide*.
- **Native UltraLite for Java** See **ianywhere.native_ultralite.SyncParms** and **ianywhere.native_ultralite.Connection.syncParms** in the API Reference.

Synchronization occurs through a synchronization **stream**. The available streams are as follows.

Component	TCP/IP	HTTP	HTTPS[1]	ActiveSync	HotSync
UltraLite for eMbedded Visual Basic	✓	✓	✓		
UltraLite for MobileVB	✓	✓			✓
Native UltraLite for Java	✓	✓	✓	✓	

[1] Requires separately licensable component

Regardless of the stream, you control the synchronization process using the same SQL scripts defined in your consolidated database or Java methods.

Secure synchronization

To synchronize using encrypted synchronization (HTTPS) you must obtain the separately-licensable security option. To order this option, see the card in your SQL Anywhere Studio package or see http://www.sybase.com/detail?id=1015780.

User authentication

UltraLite provides a built-in scheme to authenticate users before allowing them to connect to the UltraLite database.

When an UltraLite database is created, it has an initial user ID of DBA, with a password of SQL. These are also default values on connection parameters. You can avoid user authentication by not supplying **uid** or **pwd** connection parameters when connecting.

UltraLite permits up to four different users to be defined at a time, with both user ID and password being less than 16 characters long.

Each user has full access to the database once successfully authenticated. The user authentication scheme does not provide the permissions features implemented in multi-user database systems and in MobiLink synchronization.

The case sensitivity of UltraLite databases is set when the database schema is created. If the database is case insensitive (the default) then the user ID and password are case insensitive. If the database is case sensitive, then the password is case sensitive.

❖ **To user authentication to your application:**

1. Connect to the database using the default **uid** and **pwd** parameters.

   New users have to be added from an existing connection.

2. Prompt for a user ID and password.
3. Grant access to this user.

   Use the ULConnection.GrantConnectTo method to enable user authentication and provide access to a specific user ID and password combination.

4. Optionally, revoke access from the original user ID.

CHAPTER 2

Utility Programs

About this chapter

This chapter provides reference information about the UltraLite Component Suite utility programs. The following utilities are required to build applications built with the UltraLite Component Suite.

Contents

Introduction to UltraLite Component Suite utilities

The database **schema** is the database without the data. It is the collection of tables, indexes, and so on within the database, and all the relationships between them. The **schema file** stores schema information. You do not alter the schema of an UltraLite database directly. Instead, you modify a schema file (which typically has the extension *.usm*) and upgrade the database schema from that file using a built-in UltraLite function in your application.

You can create an UltraLite schema file in the following ways:

- **Generate the schema from an Adaptive Server Anywhere database** If you have the Adaptive Server Anywhere database management system, you can generate an UltraLite schema file using the *ulinit* command line utility.
- **Schema Painter** The UltraLite Schema Painter is a graphical utility for creating and editing UltraLite schema files.

  To start the Schema Painter, choose Start➤Programs➤Sybase SQL Anywhere 8➤UltraLite Schema Painter, or double-click a schema file (with extension *usm*) in Windows Explorer.
- **The ulxml command line utility** The ulxml command line utility allows you to open usm files and save them to xml format, open xml files and save them as usm files, and to export XML files to a format suitable for Palm. For more information, see "The ULXML command line utility" on page 22.

UltraLite initialization utility

Function

The *ulinit* utility lets you create a *.usm* file for use with any UltraLite component. The utility connects to an Adaptive Server Anywhere database. Consequently, SQL Anywhere Studio (version 8.0.2) is *required* in order to use it.

Syntax

ulinit -f *schema_file* **-n** *pub_name* [*options*]

Option	Description
-c *"connection_string"*	Supply database connection parameters in the form *keyword=value*, separated by semi-colons. You supply these so you may connect to an Adaptive Server Anywhere database.
-f <*schema_file* >	Specify the name of the output file. This option is required.
-m <*version*>	Specify the version string for generated MobiLink scripts.
-n <*pubname*>	Generate schema for the named publication. Specify the switch multiple times for multiple publications. Use * to generate schema for all tables in the database. For example, **-n***. This option is required.
-o *"keyword=value;..."*	Supply schema creation options.
-palm <*id*>	Create a schema file compatible with PalmOS. Id is the four digit Palm creator id that identifies the database.
-q	Quiet operation — only report errors and warnings.
-s <*pubname*>	Specify a publication name for synchronization. This option can be used multiple times.
-t <*file*>	Specify the file containing the trusted root certificates.
-w	Do not display warnings.
-z <*ordering*>	Specify table ordering (for example, *-z table1,table2*).

Examples

The following example creates a file called *customer.usm* that contains the tables in TestPublication:

```
ulinit -c "uid=dba;pwd=sql" -f customer.usm -n
TestPublication
```

When creating an UltraLite schema for Palm with *ulinit*, use the -palm switch. This generates a *.pdb* file.

```
ulinit -c "uid=dba;pwd=sql;dsn=ASA 8.0 Sample"
-f tutcustomer.usm -n TutCustomersPub -palm Syb3
```

Note
Syb3 is the four digit Palm registered creator ID that matches the creator ID of your application. For AppForge developers, this must be set in your MobileVB project settings.

The PDB file generated by *ulinit* must be loaded to the Palm device. When an UltraLite application needs to connect to the database, it should include the creator ID in the parameters of the call to Open. For example:

```
DatabaseManager.OpenConnection( "palm_db=Syb3" )
```

The UltraLite Schema Painter

The UltraLite Schema Painter allows you to create a new UltraLite schema file or edit an existing one. Thus, even if you do not have Adaptive Server Anywhere installed, or if you are unfamiliar with Adaptive Server Anywhere, you can:

- Create a new schema, or edit an existing schema
- Add, edit, or delete a new table by double-clicking Add Table
- Add, edit, and delete publications, columns, foreign keys, and indexes
- Export the schema as a *.pdb* file suitable for Palm OS devices
- Save as a *.usm* file or Open a *.usm* file for PocketPC devices

Starting the UltraLite Schema Painter

❖ **To start the UltraLite Schema Painter:**

1. Start the UltraLite Schema Painter:

   Choose Start➤Programs➤SQL Anywhere 8➤UltraLite Schema Painter.

Create, save and export schema files

❖ **To create a new schema file:**

1. Open the Tools folder and double-click Create UltraLite Schema.
2. In the New Schema dialog, type in a file name.
3. Click OK to create the schema.

❖ **To save a file:**

1. Choose File➤Save to save the file.
2. You can select to Save in *.xml*, or *.usm* format.

❖ **To export a Palm schema file:**

1. Right-click the schema icon and choose Export Schema for Palm from the popup menu.

2 Enter a Creator ID.

3 Click OK.

Managing schema files

UltraLite stores the original name, the old name, of table and column objects when you rename a table or column in your schema. This is done on the first rename. For example, if you create a table named cust, and later rename it to customer, cust is saved as the old name. If you then renamed the table a second time, to customer_info, the old name remains cust. This is done so that your schema file can be used to upgrade an existing database. For example, assume that version one of your application shipped with a table named cust. As part of the changes for version two, you modify your version one schema file by renaming the table to customer. This automatically saves cust as the old name. If you now apply this schema file to a version one database file, UltraLite will look for a table named cust, the old name, and rename it customer. The same applies to columns in a table.

You can see how it is therefore important to clear this upgrade information, the old names, after a schema file is deployed.

❖ **To clear all of the old names in the schema file after deployment:**

1 Open the schema file in the UltraLite Schema Painter

2 Right-click on the database

3 Select Clear Upgrade Information

This sets all of the old names for tables and columns to empty values. You can then safely edit your schema file for the next version of your application.

Sometimes it may be desirable to manually alter the old names of tables and columns. For example, you may have versions one and two of your application deployed and wish to create a single UltraLite schema file that can upgrade both versions one and two of this database to version three.

❖ **To manually change old names:**

1 Open your schema in the UltraLite Schema Painter

2 Right-click on the database

3 Choose "Prepare Schema for Deployment

You can use this feature to inspect the current old names in your schema. And if you are using ulxml, you can explicity set the old name of tables and columns in the <table> and <column> XML elements.

The ULXML command line utility

Function

The *ulxml* utility lets you convert data file formats. For example, you can create a *.usm* file based on an XML file. It can be used with any UltraLite component.

Syntax

ulxml [*options*] *input_file output_file*

Option	Description
-y	Overwrite output file if it already exists.
-to<*type*> *where type=xml|usm|pdb* Note: pdb files require a CreatorID.	Converts the file to one of these standard formats. Use *toxml* to convert an UltraLite schema to XML. Use *tousm* to convert an XML file to an UltraLite schema Use *topdb* to convert an XML file to an UltraLite schema for Palm.

The return code from ULXML is set to 0 on success and less than 0 on failure.

You can export your UltraLite schema so that you can work in XML format:

```
<?xml version="1.0" ?>
- <ul:ulschema xmlns:ul="urn:ultralite">
  - <tables>
    - <table name="ULCustomer">
      - <columns>
          <column name="cust_id" type="integer" null="yes"
            default="global_autoincrement" />
          <column name="cust_name" type="varchar(30)" />
        </columns>
      - <primarykey>
          <primarycolumn name="cust_id" direction="asc" />
        </primarykey>
      - <indexes>
        - <index name="ULCustomerName" unique="yes">
            <indexcolumn name="cust_name" direction="asc" />
          </index>
        </indexes>
      </table>
    + <table name="ULProduct">
    - <table name="ULEmployee">
      - <columns>
          <column name="emp_id" type="integer" null="no" />
```

You can view and use the documented sample located in *Samples\NativeUltraLiteForJava\sample.xml*, *Samples\UltraLiteActiveX\sample.xml*, and *Samples\UltraLiteForMobileVB\sample.xml*.

Note
The UltraLite Schema Painter by default creates, opens and saves UltraLite schema files in their native *.usm* file format. However, you are given the option to create, open and save XML files as well by choosing UltraLite XML Schema Files in any file type dropdown box.

CHAPTER 3

Connection Parameters

About this chapter

This chapter provides a reference for the parameters that establish and describe connections from client applications to a database.

Contents

Summary

Connection parameters are needed for your applications to work with a specific schema file, to create or connect to a database with a specified name and location, to adjust cache size, and to tune connection characteristics.

Connection parameters are case insensitive.

The following table lists the available connection parameters.

Parameter	Description
cache_size	Defines the size of the cache. See "cache_size connection parameter " on page 35
ce_file	The path and filename of the UltraLite database file to which you want to connect on Windows CE. See "ce_file connection parameter" on page 30.
ce_schema	The path and filename of the UltraLite schema on Windows CE. See "ce_schema connection parameter " on page 33
file_name	The path and filename of the UltraLite database file to which you want to connect. See "file_name connection parameter [DBF]" on page 30.
key	An encryption key for the database. See "key connection parameter " on page 35
page_size	The database page size. See "page_size connection parameter " on page 36
palm_fs	Identifies the Palm card as using the virtual file system. See "palm_fs parameter " on page 31.
palm_schema	The UltraLite schema for the Palm OS. See "palm_schema connection parameter " on page 34.
password	A password for the user. See "Password connection parameter [PWD]" on page 38.
reserve_size	Defines the reserve size. See "reserve_size connection parameter " on page 36
schema_file	The path and filename of the UltraLite schema. See "schema_file connection parameter " on page 33
userid	The user ID with which you connect to the database. See "Userid connection parameter [UID]" on page 38.

Required connection parameters

The required connection parameters for any UltraLite component depend on what methods you are using in your code. Only a schema file is required for creating a database. For opening a connection, there are no required connection parameters, although in almost all cases you need to supply the parameters for naming your database.

CreateDatabase method

The following are the basic parameters when using CreateDatabase. Required parameters are highlighted.

OS	Parameters
Windows CE	(CE_FILE or FILE_NAME or DBF) and **CE_SCHEMA** or **SCHEMA_FILE**
Windows	(FILE_NAME or DBF) and **SCHEMA_FILE**
Palm	PALM_DB and **PALM_SCHEMA**
	PALM_FS=VFS is required when using a database stored on the virtual file system.

OpenConnection method

The following are basic parameters when using OpenConnection.

OS	Parameters
Windows CE	CE_FILE or FILE_NAME or DBF
Windows	FILE_NAME or DBF
Palm	PALM_DB
	PALM_FS=VFS is required when using the virtual file system if you want the database saved on the card device.

Schema files and database files defined

Files such as CE_SCHEMA, SCHEMA_FILE, and PALM_SCHEMA are schema files. Schema files are created using either ULINIT or the Schema Painter. These files contain the database schema, or structure, that you want for your application. Database files such as ce_file, FILE_NAME, DBF or PALM_DB are the files you will use to store your data.

Platform variations for specifying file paths

File names and paths in connection parameters are subject to the following requirements, depending on the UltraLite Component you are using:

Component	Requirement
Java	All backslashes must be escaped. For example, `"file_name=\\UltraLite\\MyFile.udb"`.
Windows CE	Paths are absolute.
Windows	Paths may be absolute or relative.

Database identification parameters

The following are used to identify the UltraLite database.

ce_file connection parameter

Function The path and filename of the UltraLite database file to which you want to connect on Windows CE. Overrides the generic platform parameter file_name.

Values *String*

Default The default extension for CE files is *.udb*. ce_file is required to use a database with any name other than the default.

For more information on ce_file for the CreateDatabase method, see "CreateDatabase method" on page 27.

For more information on ce_filefor the OpenDatabase method, see "OpenConnection method" on page 27.

Description To create and connect to a specific database file.

- If the filename does not include an extension, the file of extension *.udb* is presumed.
- The path of the file is relative to the root directory.
- The schema file is not required if a *.udb* file already exists.

Example

- To create and connect to the sample database, *udemo.udb*:

```
"schema_file=MyOrders.usm;CE_FILE=udemo.udb"
```

file_name connection parameter [DBF]

Function The database file to which you want to connect. Another alias for file_name is DBF.

Values *String*

Default

Platform	Default file name
Windows:	*ulstore.udb*
Windows CE:	*\UltraLiteDB\ulstore.udb*
Palm OS	The default creator ID is the creator ID of the application.

Description

To create and connect to a specific database file.

- If a database is loaded with a name that is the same as the file_name DBF connection parameter, the connection is made to that database.
- If the filename does not include an extension, the file of extension *.udb* is presumed.
- The path of the file is relative to the working directory of the database server. If you start the server from a command prompt, the working directory is the directory that you are in when entering the command. If you start the server from an icon or shortcut, it is the working directory that the icon or shortcut specifies.

For more information on file_name for the CreateDatabase method, see "CreateDatabase method" on page 27.

For more information on file_name for the OpenDatabase method, see "OpenConnection method" on page 27.

Example

- To create and connect to the sample database, *udemo.udb*, installed in the directory *c:\Program Files\Sybase\SQL Anywhere 8*, use the following **DBF** connection parameter:

```
"schema_file=MyOrders.usm;DBF=udemo.udb"
```

palm_fs parameter

Function

Identifies the Palm card as using the virtual file system.

PALM_FS=VFS

Description

The `palm_fs=vfs` parameter needs to be specified both for CreateDatabase and OpenConnection if you are using the VFS card for Palm devices and you want the database stored on the card.

The following discussion refers to “connection parameters” but these parameters are applicable to the DropDatabase and CreateDatabase methods as well as OpenConnection.

To create, drop or connect to a database on a memory card, the following connection parameter must be specified in the parameter string:

```
Palm_fs=vfs
```

It is still possible to control the filename of the UltraLite database when it is on the card. If the file_name parameter is specified, then the database is created (or dropped or a connection attempted) on the card with the filename specified. If the file_name parameter is not specified but the palm_db parameter is, then the database created (or dropped or to which a connection is attempted) resides on the card with a filename `ul_udb_XXXX.udb` (where XXXX is the creator ID specified by the palm_db parameter). If neither parameter is specified, the filename for the database is `ul_udb_YYYY.udb` where YYYY is the creator ID of the application.

If the Palm_fs parameter is not specified, the database is created (or dropped from or to which you are connecting) on the device and not the card.

For more information on palm_fs for the CreateDatabase method, see "CreateDatabase method" on page 27.

For more information on palm_fs for the OpenDatabase method, see "OpenConnection method" on page 27.

Database schema parameters

The following keywords are used to specify a schema for an UltraLite database. Thus, schema parameters are vital database creation parameters, as your schema determines which tables and columns exist in your database. Only one file value is used with the platform specific keyword taking precedence over the generic keyword.

ce_schema connection parameter

Function To identify the schema on Windows CE.

Values *String*

Default The recommended file extension is *.usm*.

Description The path and filename of the UltraLite schema on Windows CE. The default extension for UltraLite schema files is *.usm*. ce_schema is a required parameter when using CreateDatabase for CE.

Example

- The following connection string fragment supplies the ce_schema and schema_file parameters.

```
"CE_SCHEMA=orders.usm;SCHEMA_FILE=MyOrders.usm"
```

For more information on ce_schema for the CreateDatabase method, see "CreateDatabase method" on page 27.

For more information on ce_schema for the OpenDatabase method, see "OpenConnection method" on page 27.

schema_file connection parameter

Function To identify the schema.

Values *String*

Default The recommended file extension is *.usm*.

Description The path and filename of the UltraLite schema.

For more information on schema_file for the CreateDatabase method, see "CreateDatabase method" on page 27.

For more information on schema_file for the OpenDatabase method, see "OpenConnection method" on page 27.

Example

- The following connection string fragment supplies the ce_schema and schema_file parameters.

```
"CE_SCHEMA=orders.usm;SCHEMA_FILE=MyOrders.usm"
```

palm_schema connection parameter

Function To identify the schema for Palm.

Values *String*

Default The Palm file extension on the desktop is *.pdb*.

Description The filename of the UltraLite schema for Palm. The palm_schema parameter is a required parameter when using CreateDatabase on Palm devices.

Although *.pdb* is the extension on the desktop, do not supply *.pdb* in your connection parameter string.

For more information on palm_schema for the CreateDatabase method, see "CreateDatabase method" on page 27.

For more information on palm_schema for the OpenDatabase method, see "OpenConnection method" on page 27.

Example

- The following connection string fragment supplies the palm_schema and schema_file parameters.

```
"PALM_SCHEMA=orders;SCHEMA_FILE=MyOrders.usm"
```

Other database creation parameters

Database creation parameters are optional parameters to configure a database when it is created. Some of these parameters can influence performance, so it is suggested that you test these parameters to find the optimal performance for your application.

cache_size connection parameter

Function Defines the size of the cache.

Usage Used when you configure a database. Use `k` or `K`, `m` or `M` to denote kilobytes or megabytes, respectively.

Values The minimum cache size is 4K.

Default The default is 16 x page_size. Actual value used is rounded down to the nearest multiple of page_size.

Description Defines the size of the cache. You can specify the size in units of bytes. Use the suffix k or K to indicate units of kilobytes and use the suffix M or m to indicate megabytes

The default cache size is sixteen pages. Using the default page size of 4 K, the default cache size is therefore 64 K. The minimum cache size is platform dependent.

The default cache size is conservative. If your testing shows the need for better performance, you should increase the cache size.

Increasing the cache size beyond the size of the database itself provides no performance improvement. Also, large cache sizes may interfere with the number of other applications you can use.

On the Palm Computing Platform, the parameter applies only to virtual file system (VFS) databases. The cache itself resides in record storage, not VFS storage.

Example For example, the following string sets the cache size to 128 K.

```
"cache_size=128k"
```

key connection parameter

Function An encryption key for the database. You can define an encryption key for your UltraLite database when CreateDatabase is called.

Values *String*

Default No key is provided.

Description If a database is created using key, UltraLite database files are strongly encrypted using the AES 128-bit algorithm, which is the same algorithm used to encrypt Adaptive Server Anywhere databases. Use of strong encryption does provide security against skilled and determined attempts to gain access to the data, and has a significant performance impact.

Example

```
"schema_file=MyOrders.usm;KEY=MyKey"
```

page_size connection parameter

Function Defines the paging size.

Usage Used when you configure a database. Use `k` or `K` to denote kilobytes.

Default The default page size for UltraLite databases is 4 K. The range of size is 2 K to 4 K.

Description UltraLite databases are stored in pages. I/O operations are carried out a page at a time. It can be used on any target platform. Setting a page size of 2 K reduces the maximum number of tables to approximately 500.

Example You can specify 2 kb pages using the following storage parameters string:

```
"schema_file=MyOrders.usm;PAGE_SIZE=2K"
```

reserve_size connection parameter

Function Defines the reserve size.

Usage Used when you configure a database. Use `k` or `K`, `m` or `M` to denote kilobytes or megabytes, respectively.

Values Values can be expressed in kb or mb.

Description The reserve_size parameter allows you to pre-allocate the file system space required for your UltraLite database without actually inserting any data. Reserving file system space can improve performance slightly and also prevent out of memory failures. By default, the persistent storage file only grows when required as the application updates the database.

Reserve_size reserves file system space, which includes the metadata in the persistent store file, and not just the raw data. The metadata overhead as well as data compression must be considered when deriving the required file system space from the amount of database data. Running the database with test data and observing the persistent store file size is recommended.

The reserve_size parameter reserves space by growing the persistent store file to the given reserve size on startup, regardless of whether the file previously existed. The file is never truncated.

Example

Use the reserve_size parameter to pre-allocate space as follows:

```
"CE_SCHEMA=orders;RESERVE_SIZE=128K"
```

This example ensures that the persistent store file is at least 128 K upon startup.

User authentication parameters

User authentication parameters are used to identify the user as authorized to use the database.

Password connection parameter [PWD]

Function A password for the user. Passwords are case insensitive if the database is case insensitive and case sensitive if the database is case sensitive.

Usage Anywhere

Values *String*

Default SQL

Description Every user of a database has a password. The password must be supplied for the user to be allowed to connect to the database.

The Password (PWD) connection parameter is not encrypted.

Example

- The following connection string fragment supplies the user ID DBA and password SQL.

```
"UID=DBA;PWD=SQL;schema_file=MyOrders.usm"
```

Userid connection parameter [UID]

Function The user ID with which you log on to the database. An authenticated user for the database. User ID's are case-insensitive if the database is case-insensitive and case sensitive if the database is case sensitive.

Databases are created with a single authenticated user DBA whose initial password is SQL. By default, connections are opened using the UID=DBA and the PWD=SQL. To disable the default user, use

`connection.revokeConnectionFrom`.

To add a user or change a user's password, use

`connection.grantConnectTo`.

Usage Anywhere

Values *String*

Default DBA

Description

You must always supply a userID when connecting to a database, unless you leave the database using the default user ID and password of DBA and SQL.

Example

- The following connection string fragment supplies the user ID DBA and password SQL:

```
"schema_file=MyOrders.usm;uid=DBA;pwd=SQL"
```

CHAPTER 4

Synchronization Stream Parameters Reference

About this chapter

This chapter lists the stream parameters for each synchronization stream. This chapter is intended for users who have SQL Anywhere Studio.

Contents

Introduction

UltraLite databases can synchronize with a MobiLink synchronization server over one of a set of synchronization streams, including TCP/IP, HTTP, and HotSync for Palm OS applications. Each synchronization stream has a set of appropriate stream parameters. These parameters set required values for the stream, such as the location of the MobiLink synchronization server, and network-specific control parameters. This chapter lists the stream parameter values for each stream.

Meaning differs for HotSync and ActiveSync
For HotSync and ActiveSync synchronization, the meaning of the synchronization stream parameters is different than for other streams. For information, see "HotSync synchronization stream parameters" on page 45 and "ActiveSync synchronization stream parameters" on page 43.

Setting a stream

The way to select a synchronization stream depends on the component you are using.

- For MobileVB and eMbedded Visual Basic applications, the synchronization stream is one of the synchonization parameters set in the Stream property of the ULSyncParms object. The stream parameters are provided as a set of keyword-value pairs in the StreamParms property.

  For more information, see "ULSyncParms class" on page 93 of the book *UltraLite for AppForge User's Guide*, and "ULSyncParms class" on page 84 of the book *UltraLite for eMbedded Visual Basic User's Guide*.

- For Native UltraLite for Java applications, the synchronization stream is set by the **setStream** method of the **SyncParms** object.

  For more information, see **ianywhere.native_ultralite.SyncParms** in the API Reference.

ActiveSync synchronization stream parameters

The ActiveSync synchronization stream is accessible only from Native UltraLite for Java applications running on Windows CE.

To choose ActiveSync synchronization, supply StreamType.ACTIVE_SYNC as the argument to the syncParms.setStream method.

For more information, see **ianywhere.native_ultralite.StreamType** and **ianywhere.native_ultralite.SyncParms** in the Native UltraLite for Java API Reference.

Example

The following line sets ActiveSync as the synchronization stream:

```
_conn.syncParms.setStream( StreamType.ACTIVE_SYNC );
```

Meaning of synchronization stream parameters

The stream parameters control the connection from the MobiLink ActiveSync provider, running on the desktop machine, to the MobiLink synchronization server.

The stream parameters take the following form:

```
{stream=stream_name;provider_stream_parameters}
```

where *stream_name* indicates the protocol for the conduit to use when communicating from the conduit to the MobiLink synchronization server. It must be one of the following:

- **tcpip**
- **http**
- **https**

and where *provider_stream_parameters* is a set of stream parameters for use by the ActiveSync provider, and has the same form as the stream parameters for the protocol in use. For the given stream, the *provider_stream_parameters* adopts the same defaults as the stream parameters for the protocol. The default value for the *stream_name* is **tcpip**.

For more information on *provider_stream_parameters*, see "TCP/IP stream parameters" on page 47, and "HTTP stream parameters" on page 49.

Adding encryption to ActiveSync synchronization

To add Certicom encryption to the stream, the root certificates must be in a file on the desktop machine. This is different from other UltraLite applications, where the encryption information is embedded in the **security** synchronization parameter.

The stream parameters need to be specified in the stream parameters in much the same way as for Adaptive Server Anywhere MobiLink clients . The format is:

security=*cipher*{ *keyword*=*value*;... }

where *cipher* must be certicom_tls and the keywords are taken from the following list:

- **certificate_company** The organization field on the certificate.
- **certificate_unit** The organization unit field on the certificate.
- **certificate_name** The common name field on the certificate.
- **trusted_certificates** The location of the trusted certificates.

HotSync synchronization stream parameters

The HotSync synchronization stream is accessible only from UltraLite for MobileVB applications running on the Palm Computing Platform.

To choose HotSync synchronization, choose ulPalmConduit from the ULStreamType enumeration as the ULSyncParms.Stream.

☞ For more information, see "ULSyncParms class" on page 93 of the book *UltraLite for AppForge User's Guide*.

Meaning of synchronization stream parameters

For HotSync synchronization, the stream parameters do *not* control the connection from the device to the HotSync Manager or HotSync Server. Instead, they specify the connection from the MobiLink conduit, running at the HotSync manager or server, to the MobiLink synchronization server.

The argument has the following form:

```
{stream=stream_name;conduit_stream_parameters}
```

where *stream_name* indicates the protocol for the conduit to use when communicating from the conduit to the MobiLink synchronization server. It must be one of the following:

- **tcpip**
- **http**

and where *conduit_stream_parameters* is a set of stream parameters for use by the conduit, and has the same form as the **stream_parms** argument for the protocol in use. For the given stream, the *conduit_stream_parameters* adopts the same defaults as the **stream_parms** argument for the protocol. The default value for the *stream_name* is tcpip.

☞ For more information on *conduit_stream_parameters*, see "TCP/IP stream parameters" on page 47, and "HTTP stream parameters" on page 49.

Null value and default settings

If you use HotSync synchronization, and do not supply stream parameters, the conduit searches in the registry for the stream name and stream parameters. If it finds no valid stream, the default stream and stream parameters is used. This default stream parameter setting is:

```
{stream=tcpip;host=localhost}
```

Adding encryption to HotSync synchronization

To add Certicom encryption to the stream, the root certificates must be in a file on the desktop machine. This is different from other UltraLite applications, where the encryption information is embedded in the **security** synchronization parameter.

The stream parameters need to be specified in the stream parameters in much the same way as for Adaptive Server Anywhere MobiLink clients . The format is:

security=*cipher*{ *keyword*=*value*;... }

where *cipher* must be certicom_tls and the keywords are taken from the following list:

- **certificate_company** The organization field on the certificate.
- **certificate_unit** The organization unit field on the certificate.
- **certificate_name** The common name field on the certificate.
- **trusted_certificates** The location of the trusted certificates.

TCP/IP stream parameters

The TCP/IP synchronization stream is accessible from all UltraLite components. To select TCP/IP as the synchronization stream:

- In UltraLite for MobileVB and UltraLite for eMbedded Visual Basic, choose ulTCPIP from the ULStreamType enumeration as the ULSyncParms.Stream.

  For more information, see "ULSyncParms class" on page 93 of the book *UltraLite for AppForge User's Guide*.

- In Native UltraLite for Java, supply StreamType.TCPIP as the argument for SyncParms.setStream().

  For more information, see **ianywhere.native_ultralite.StreamType** and **ianywhere.native_ultralite.SyncParms** in the Native UltraLite for Java API Reference.

Synchronization stream parameters for the TCP/IP stream are chosen from the following table:

Parameter	Description
client_port=*nnnnn* **client_port**=*nnnnn-mmmmm*	A range of client ports for communication. If only one value is specified, the end of the range is 100 greater than the initial value, for a total of 101 ports. The option can be useful for clients inside a firewall communicating with a MobiLink synchronization server outside the firewall.
host=*hostname*	The host name or IP number for the machine on which the MobiLink synchronization server is running. The default value is **localhost**, except on Windows CE. For Windows CE, the default setting corresponds to the desktop machine where the CE device's cradle is connected, which is stored as the *ipaddr* entry in the registry folder *Comm\Tcpip\Hosts\ppp_peer*. Do not use **localhost**, which refers to the device itself, on Windows CE. For the Palm Computing Platform, the default value of **localhost** refers to the device itself. You should supply an explicit host name or IP address to connect to a desktop machine.
port=*portnumber*	The socket port number on the host machine. The port number must be a decimal number that matches the port the MobiLink synchronization server is setup to monitor.

Parameter	Description
	The default value for the port parameter is 2439, which is the IANA registered port number for the MobiLink synchronization server.

HTTP stream parameters

The HTTP synchronization stream is accessible from all UltraLite componetns. To select HTTP as the synchronization stream:

- In UltraLite for MobileVB and UltraLite for eMbedded Visual Basic, choose ulHTTP from the ULStreamType enumeration as the ULSyncParms.Stream.

  For more information, see "ULSyncParms class" on page 93 of the book *UltraLite for AppForge User's Guide*.

- In Native UltraLite for Java, supply StreamType.HTTP as the argument for SyncParms.setStream().

  For more information, see **ianywhere.native_ultralite.StreamType** and **ianywhere.native_ultralite.SyncParms** in the Native UltraLite for Java API Reference.

Synchronization stream parameters for the HTTP stream are chosen from the following table:

Parameter	Description
client_port=*nnnnn* **client_port**=*nnnnn-mmmmm*	A range of client ports for communication. If only one value is specified, the end of the range is 100 greater than the initial value, for a total of 101 ports. The option can be useful for clients inside a firewall communicating with a MobiLink synchronization server outside the firewall.
version=*versionnumber*	A string specifying the version of HTTP to use. You have a choice of **1.0** or **1.1**. The default value is **1.1**.
host=*hostname*	The host name or IP number for the machine on which the MobiLink synchronization server is running. The default value is **localhost**. For Windows CE, the default value is the value of *ipaddr* in the registry folder *Comm\Tcpip\Hosts\ppp_peer*. This allows a CE device to connect to a MobiLink synchronization server executing on the desktop machine where the CE device's cradle is connected. For the Palm Computing Platform, the default value of **localhost** refers to the device. It is recommended that an explicit host name or IP address be specified.
port=*portnumber*	The socket port number. The port number must be a decimal number that matches the port the MobiLink synchronization server is setup to monitor. The default value for the port parameter is 2439, which is the IANA registered port number for the MobiLink synchronization server.
proxy_host=*proxy_hostname*	The host name of the proxy server.
proxy_port=*proxy_portnumber*	The port number of the proxy server. The default value is 80.
url_suffix=*suffix*	The suffix to add to the URL on the first line of each HTTP request. When synchronizing through a proxy server, the suffix may be necessary in order to find the MobiLink synchronization server. The default value is **MobiLink**.
use_cookies	Control sessions when synchronizing through a Web server using HTTP. Set this parameter to **ul_true** when

Parameter	Description
	synchronizing through a Web server. The default value is **ul_false** (0).

HTTPS stream parameters

The HTTPS synchronization stream is accessible from all UltraLite components.

❖ **To select HTTPS as the synchronization stream:**

- In UltraLite for MobileVB and UltraLite for eMbedded Visual Basic, choose ulHTTPS from the ULStreamType enumeration as the ULSyncParms.Stream.

  For more information, see "ULSyncParms class" on page 93 of the book *UltraLite for AppForge User's Guide*.

- In Native UltraLite for Java, supply StreamType.HTTPS as the argument for SyncParms.setStream().

  For more information, see **ianywhere.native_ultralite.StreamType** and **ianywhere.native_ultralite.SyncParms** in the Native UltraLite for Java API Reference.

Separately-licensable option required
Use of Certicom technology requires that you obtain the separately-licensable SQL Anywhere Studio security option and is subject to export regulations. For more information on this option, see "Welcome to SQL Anywhere Studio" in the book *Introducing SQL Anywhere Studio*.

Synchronization stream parameters for the HTTPS stream are chosen from the following table:

Parameter	Description
client_port=*nnnnn* **client_port**=*nnnnn-mmmmm*	A range of client ports for communication. If only one value is specified, the end of the range is 100 greater than the initial value, for a total of 101 ports. The option can be useful for clients inside a firewall communicating with a MobiLink synchronization server outside the firewall.
host=*hostname*	The host name or IP number for the machine on which the MobiLink synchronization server is running. The default value is **localhost**. For Windows CE, the default value is the value of *ipaddr* in the registry folder *Comm\Tcpip\Hosts\ppp_peer*. This allows a CE device to connect to a MobiLink synchronization server executing on the desktop machine where the CE device's cradle is connected. For the Palm Computing Platform, the default value of **localhost** refers to the device. It is recommended that an explicit host name or IP address be specified.
port=*portnumber*	The socket port number. The port number must be a decimal number that matches the port the MobiLink synchronization server is setup to monitor. The default value for the port parameter is 2439, which is the IANA registered port number for the MobiLink synchronization server.
proxy_host= *proxy_hostname*	The host name of the proxy server.
proxy_port= *proxy_portnumber*	The port number of the proxy server. The default value is 80.
certificate_company	The UltraLite application only accepts server certificates when the organization field on the certificate matches this value. By default, this field is not checked.

Parameter	Description
certificate_name	The UltraLite application only accepts server certificates when the common name field on the certificate matches this value. By default, this field is not checked.
certificate_unit	The UltraLite application only accepts server certificates when the organization unit field on the certificate matches this value. By default, this field is not checked.
url_suffix=*suffix*	The suffix to add to the URL on the first line of each HTTP request. When synchronizing through a proxy server, the suffix may be necessary in order to find the MobiLink synchronization server. The default value is **MobiLink**.
use_cookies	Control sessions when synchronizing through a Web server using HTTP. Set this parameter to **ul_true** when synchronizing through a Web server. The default value is **ul_false** (0).
version=*versionnumber*	A string specifying the version of HTTP to use. You have a choice of **1.0** or **1.1**. The default value is **1.1**.

Index

F

H

I

K

M

N

P

R

S

T

U

V