Oracle® Database Lite

SQLite Mobile Client Guide

Release 10.3

E16214-01

February 2010

ORACLE®

Oracle Database Lite SQLite Mobile Client Guide Release 10.3

E16214-01

Contents

Preface v

1 Overview of The SQLite Mobile Client

1.1 SQLite Database Overview 1-1
1.2 SQLite Mobile Client Architecture 1-1

2 Installing the SQLite Mobile Client

2.1 Supported Platforms and Requirements for the SQLite Mobile Client 2-1
2.1.1 Certified Operating Systems and Other Software Requirements 2-2
2.1.2 Supported and Certified Technologies for Native Mobile Clients 2-3
2.2 Preparing the Device for a Mobile Application 2-4
2.3 Installing the SQLite Mobile Client 2-5
2.3.1 Installing the SQLite Mobile Client on Blackberry Devices 2-5
2.3.2 Installing the SQLite Mobile Client on Android Devices 2-6
2.3.3 Installing the SQLite Mobile Client for Win32, WinCE, Windows Mobile or Linux. 2-6
2.3.3.1 Installing Standard SDK WinCE 5.0 CAB Files for Your Mobile Client 2-9
2.4 Configuring the Location of the SQLite Database and Mobile Client Files 2-12
2.5 Configuring for Automatic Synchronization When Installing the Client 2-12

3 Managing Your SQLite Mobile Client

3.1 Starting the SQLite Mobile Client 3-1
3.2 Synchronize Data for Applications on the SQLite Mobile Client 3-1
3.3 Use the mSync GUI to Initiate Synchronization 3-2
3.3.1 Network Options for MSync Tool 3-3
3.3.2 Sync Options for MSync Tool 3-4
3.3.3 Sync to a File Using File-Based Sync 3-5
3.3.4 Use SQLite Mobile Client Tools on Linux 3-5
3.4 Reset the Mobile User Password 3-5
3.5 Manage Snapshots on SQLite Mobile Client 3-6
3.6 Control Automatic Synchronization for a Specific SQLite Mobile Client 3-7
3.7 Improve Performance by Disabling the Resume Feature 3-7
3.8 Use the Device Manager Client GUI to Manage the Client-Side Device 3-7
3.9 Initiate Updates for the SQLite Mobile Client 3-7
3.10 Communicate Between the Internet and Intranet Through a Reverse Proxy 3-8

4 Using an Android Application on the SQLite Mobile Client

4.1 Prerequisites 4-1
4.2 Developing an Android Application for Oracle Database Lite........ 4-2
4.3 Import the Oracle Database Lite Android Project into Eclipse 4-2
4.4 Build Oracle Database Lite Android Project........ 4-3

A INI Configuration Parameters

A.1 OSE.INI File Overview........ A-1
A.1.1 SQLITE.DATA_DIRECTORY A-2
A.1.2 SQLITE.QUEUES........ A-2
A.1.3 OSE.RESUME........ A-3
A.1.4 BGSYNC.DISABLE........ A-3
A.2 POLITE.INI File Overview A-3

Index

Preface

This preface introduces you to the *Oracle Database Lite SQLite Mobile Client Guide* discussing the intended audience, documentation accessibility, and structure of this document.

Audience

This manual is intended for application developers as the primary audience and for database administrators who are interested in application development as the secondary audience.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at `http://www.fcc.gov/cgb/consumerfacts/trs.html`, and a list of phone numbers is available at `http://www.fcc.gov/cgb/dro/trsphonebk.html`.

Send Us Your Comments

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: olitedoc_us@oracle.com
- FAX: (650) 506-7355. Attn: Oracle Database Lite
- Postal service:

  Oracle Corporation
  Oracle Database Lite Documentation
  500 Oracle Parkway, Mailstop 4op2
  Redwood Shores, CA 94065
  U.S.A.

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

1

Overview of The SQLite Mobile Client

The following sections describe the SQLite Mobile client:

- Section 1.1, "SQLite Database Overview"
- Section 1.2, "SQLite Mobile Client Architecture"

1.1 SQLite Database Overview

The SQLite database is a small, compact, and self-contained database available on multiple platforms and available to the public. It has a small footprint and is easy to install and administer. In addition, many devices have the SQLite database already installed, including Android and Blackberry devices.

Oracle Database Lite offers you the ability to synchronize the data entered in one or more SQLite databases to a back-end Oracle database with the SQLite Mobile client. The SQLite database is independent from the SQLite Mobile client. The SQLite Mobile client provides the ability to synchronize the data in SQLite databases with the Sync Engine. This book describes how to configure, manage and implement synchronization using the Sync Engine. It does not discuss how to configure, manage or use the SQLite database. For information on the SQLite database itself, see `http://www.sqlite.org/`.

The SQLite database does not provide the same SQL functionality as Oracle databases. Thus, refer to `http://www.sqlite.org/` for a full list of what is supported.

The SQLite Mobile client can be installed on the following platforms: Linux, Windows (Win32), WinCE, Android and Blackberry platforms. Device management is supported on Win32, WinCE and Linux platforms. The Sync Engine supports both automatic and manual synchronization for SQLite databases. However, without device management support, remote device management and automatic synchronization is not supported on Android and Blackberry platforms.

1.2 SQLite Mobile Client Architecture

As shown in Figure 1–1, when both the SQLite database and the SQLite Mobile client are installed, the mobile device has the following components:

- SQLite database—A small database that is installed independently.
- SQLite Mobile client—When you install the SQLite Mobile client, the following components are provided:
  - Sync Engine—Enables manual or automatic synchronization. Manual synchronization is initiated by the mobile application. Automatic

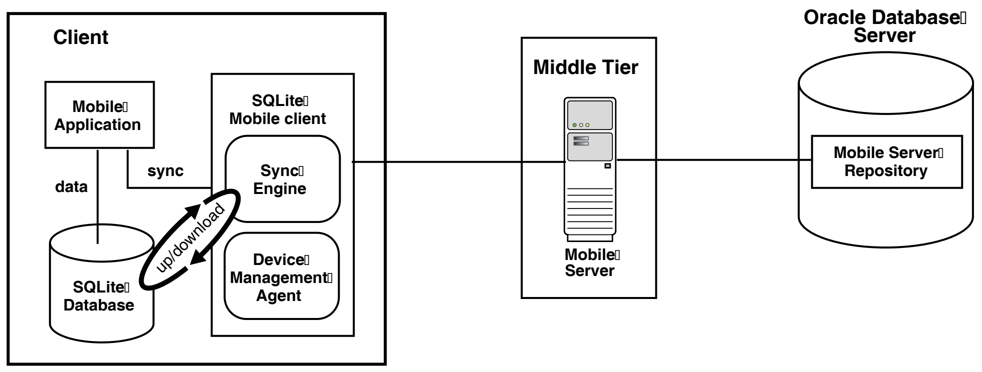
synchronization can be enabled only on the Win32, WinCE, and Linux platforms.

The Sync Engine interacts with the SQLite database to upload and download data in conjunction with the Mobile Server to synchronize the data with the Oracle database.

– Device Manager Agent (DM Agent)—The DM Agent is only installed on Win32, WinCE, and Linux. The Mobile Server uses the DM Agent to send commands to the device and manage the mobile device. The Blackberry and Android platforms cannot be managed by the Mobile Server, so the DM Agent is not installed on these devices.

- Mobile application—Interacts with the SQLite database to manage the data. Interacts with the Sync Engine to initiate a manual synchronization.

Figure 1–1 Architecture for Client with SQLite Database and SQLite Mobile Client



2 Installing the SQLite Mobile Client

One of the benefits of Oracle Database Lite is that you can have an application downloaded onto a device, where data can be synchronized between the device and the back-end Oracle database. When you install the SQLite Mobile client, Oracle Database Lite installs the Sync Engine.

The following sections detail how to install the SQLite Mobile client software on your client machine:

- Section 2.1, "Supported Platforms and Requirements for the SQLite Mobile Client"
- Section 2.2, "Preparing the Device for a Mobile Application"
- Section 2.3, "Installing the SQLite Mobile Client"
- Section 2.4, "Configuring the Location of the SQLite Database and Mobile Client Files"
- Section 2.5, "Configuring for Automatic Synchronization When Installing the Client"

See Chapter 1, "Oracle Database Lite Management With the Mobile Server" in the *Oracle Database Lite Administration and Deployment Guide* for information on how to manage functionality from the Mobile Server.

2.1 Supported Platforms and Requirements for the SQLite Mobile Client

In general, you can synchronize data in an installed SQLite database on the following Mobile client platforms:

- Blackberry and Android devices
- Windows clients
- Linux, Win32, and WinCE clients

Note: You can configure only one device for a particular user. For example, it is not possible to have two devices both executing the SQLite Mobile client for the user `JOHN`.

Automatic synchronization and device management are only available on certain SQLite Mobile client platforms. Table 2–1 displays what features are available on which platforms.

Table 2–1 Platform Restrictions

Platform	Automatic synchronization	Device management through the DM Agent
Blackberry	No	No
Android	No	No
Win32	Yes	Yes
WinCE	Yes	Yes
Linux	Yes	Yes

2.1.1 Certified Operating Systems and Other Software Requirements

The following tables detail the requirements for the client platforms on which you may install the SQLite Mobile client. The requirements do not include the SQLite database requirements, but are only the requirements for the SQLite Mobile client including the Sync Engine.

- Table 2–2, " BlackBerry and Android Platform Requirements"
- Table 2–3, " Software Requirements for Windows Mobile Clients"
- Table 2–4, " Supported and Certified Technologies for Native Mobile Clients"
- Table 2–5, " Pocket PC and Windows Mobile Supported Platforms"

Table 2–2 BlackBerry and Android Platform Requirements

Platform	Minimum OS Version	Minimum Storage for Mobile Client
BlackBerry	RIM 5.0	100 KB
Android	Android 1.5	100 KB

Table 2–3 Software Requirements for Windows Mobile Clients

Device Platform	Certified Operating System	Other Software Requirements
Win32 Minimum storage needed for Mobile client is 2,756 KB.	Windows Vista Ultimate, Windows XP Professional Edition with Service Pack 2, or Windows 2003	If using Java APIs for synchronization, use JRE 5.0 If implementing any .NET applications, use Compact Framework .NET 1.1 or 2.0
Windows CE	Windows CE 5.0 See Table 2–5, " Pocket PC and Windows Mobile Supported Platforms" for full details.	If using synchronization, use either IBMJ9 or the CrEme JDK version 4.1 from `NSI.com`. ActiveSync version 3.8 or higher. Microsoft .NET Compact Framework 1.0

Table 2–3 (Cont.) Software Requirements for Windows Mobile Clients

Device Platform	Certified Operating System	Other Software Requirements
Windows Mobile	Windows Mobile 2003 Windows Pocket PC 2003 Windows Mobile 2003, 2nd edition Windows Pocket PC 2003, 2nd edition See Table 2–5, " Pocket PC and Windows Mobile Supported Platforms" for full details.	If using synchronization, use either IBMJ9 or the CrEme JDK version 4.1 from `NSI.com`. ActiveSync version 3.8 or higher. Microsoft .NET Compact Framework 1.1
	■ Windows Mobile 5 ■ Windows Mobile 5 for Pocket PC ■ Windows Mobile 5 for Pocket PC Phone Edition ■ Windows Mobile 5 AKU2	If using synchronization, use either IBMJ9 or the CrEme JDK version 4.1 from `NSI.com`. ActiveSync version 4.1 or higher. Microsoft .NET Compact Framework 1.1 or 2.0
	■ Windows Mobile 6 ■ Windows Mobile 6 Classic ■ Windows Mobile 6 Professional	If using synchronization, use either IBMJ9 or the CrEme JDK version 4.1 from `NSI.com`. ActiveSync version 4.5 or higher. Microsoft .NET Compact Framework 1.1 or 2.0

You should install all of the patches required for the JDK for the Windows operating system. This is constantly under review and published on the JDK download page on the Sun Microsystems Web site.

2.1.2 Supported and Certified Technologies for Native Mobile Clients

The following are the supported and certified technologies for native SQLite Mobile clients:

Note: Ensure that after you install the required software, that they the appropriate directories are included in the `PATH`. For example, after you install the JDK, ensure that the `JAVA_HOME` is included in the `PATH`.

Table 2–4 Supported and Certified Technologies for Native Mobile Clients

Device Platform	Supported Technologies	Certified Technologies
Win32	■ Sun Microsystems Java Runtime Edition 5.0 ■ ADO.Net 1.1 – requires Microsoft .Net Framework 1.1 or 2.0 ■ ADO.Net 2.0 – requires Microsoft .Net Framework 2.0	
Windows CE	■ ADO.Net 1.1 (Requires Microsoft Compact .Net Framework 1.0 + Service Pack 2) or 2.0 ■ ADO.Net 2.0 – requires Microsoft .Net Compact Framework 2.0 ■ Microsoft ActiveSync version 3.8 or for Windows CE 5.0, use Microsoft ActiveSync version 4.1 or higher.	Oracle Database Lite is certified with the following JVMs on Windows Mobile 2003 Second Edition: ■ IBM J9 Websphere Everyplace Micro Environment for Windows Mobile 2003 ARM Personal Profile ■ Creme JVM 4.1, which can be obtained at `http://www.nsicom.com`
Linux x86	RedHat Enterprise Linux AS release 4	JavaSoft Java Runtime Edition 1.4.2

For each native platform, a CAB file is downloaded from the setup page. The naming structure for each CAB file is as follows: `sqlite_sync.<language>.<platform>.<cpu>.cab`

Table 2–5 Pocket PC and Windows Mobile Supported Platforms

Product Name	WinCE Version	Chipsets	SQLite Client CAB file download from Setup page
Pocket PC 2003 Windows Mobile 2003	4.20.1081	ARMV4	SQLite PPC2003 ARMV4, which uses the `sqlite_sync.<language>.ppc2003.armv4.cab`
Windows Mobile 2003 2nd Edition	4.21.1088	ARMV4	SQLite PPC2003 ARMV4, which uses the `sqlite_sync.<language>.ppc2003.armv4.cab`
Windows Mobile 5 and Windows Mobile 5 AKU2	5.0 and 5.1.465	ARMV4I	SQLite PPC50 ARMV4I, which uses the `sqlite_sync.<language>.ppc50.armv4i.cab`
Windows Mobile 6	5.2.1236	ARMV4I	SQLite PPC60 ARMV4I, which uses the `sqlite_sync.<language>.ppc60.armv4i.cab`

2.2 Preparing the Device for a Mobile Application

To execute Mobile applications on a device with the SQLite database, do the following:

Note: Install the SQLite Mobile client for any application after the application is published.

1. Install the SQLite Mobile client software that is appropriate for the client platform on your client machine. For example, install the SQLite WIN32 on a Windows 32 client machine.

   See Section 2.3, "Installing the SQLite Mobile Client" for a full description.

2. Download the user applications and its associated data.

Synchronize the Mobile client for the first time. Sign in with the username/password of the Mobile user who owns the Mobile applications. The data for each application is retrieved.

Notes: For the restrictions on creating the username and password, see Section 5.3.1.2.1, "Defining Username and Password" in the *Oracle Database Lite Administration and Deployment Guide*.

For more information about synchronization, see Chapter 6, "Managing Synchronization" in the *Oracle Database Lite Administration and Deployment Guide*.

---

3. You can now launch your applications from your client machine or from your Mobile device.

2.3 Installing the SQLite Mobile Client

We do not support the following configuration scenarios:

- A Mobile client and the Mobile Development Kit (MDK) cannot be installed on a single system.
- A client user cannot have more than one device.

The following sections provide directions for the SQLite Mobile client install:

- Section 2.3.1, "Installing the SQLite Mobile Client on Blackberry Devices"
- Section 2.3.2, "Installing the SQLite Mobile Client on Android Devices"
- Section 2.3.3, "Installing the SQLite Mobile Client for Win32, WinCE, Windows Mobile or Linux"

2.3.1 Installing the SQLite Mobile Client on Blackberry Devices

To install the SQLite Mobile client on Blackberry devices, perform the following:

Note: Applications cannot be downloaded to your Blackberry device from the Mobile Server, since device management is not supported for this device. You must download all applications to your Blackberry device as documented on the Blackberry Web site at `http://www.blackberry.com`.

---

1. On the Blackberry device, open a browser to point to the Mobile Server setup page using the following URL.

   ```
   http://<mobile_server>:<port>/webtogo/setup
   ```

   Note: Substitute `https` if using HTTP over SSL.

   Figure 2–1 displays the Mobile client setup page, which contains links to install Mobile client software for multiple languages. You can select another language than English on the Language pulldown.

2. Click the SQLite Mobile client for your language and the Blackberry client platform. This downloads and installs the SQLite Mobile client.

3. Perform a manual synchronization for the SQLite Mobile client.
4. Synchronization requires you to enter the username and password for the Mobile user. During the first synchronization, all data for this user is brought down and installed on your mobile device.

> **Note:** For the restrictions on creating the username and password, see Section 5.3.1.2.1, "Defining Username and Password" in the *Oracle Database Lite Administration and Deployment Guide*.

2.3.2 Installing the SQLite Mobile Client on Android Devices

Android platforms require that any software downloaded to the device is digitally signed with a certificate whose private key is held by the application's developer. This means that you cannot simply download and install the SQLite Mobile client binaries unless they are downloaded within the context of a signed application.

Thus, the instructions for installing the SQLite Mobile client on Android devices is described within the context of creating and downloading the Android application. These instructions are described in Chapter 4, "Using an Android Application on the SQLite Mobile Client".

2.3.3 Installing the SQLite Mobile Client for Win32, WinCE, Windows Mobile or Linux

Before you install the SQLite Mobile Client on your device, make sure that there is 1 MB of space available to download the `setup.exe`.

To install the SQLite Mobile client software, perform the following tasks.

> **Note:** Any developer can modify how the client is installed before the installation with the INF file. For details on how to customize your Win32, WinCE, or Linux SQLite Mobile client, see Section 7.1, "Customize the Mobile Client Software Installation for Your Mobile Device" in the *Oracle Database Lite Administration and Deployment Guide*.

1. On the SQLite Mobile client, open a browser to point to the Mobile Server using the following URL.

   ```
   http://<mobile_server>:<port>/webtogo/setup
   ```

   > **Note:** Substitute `https` if using HTTP over SSL.

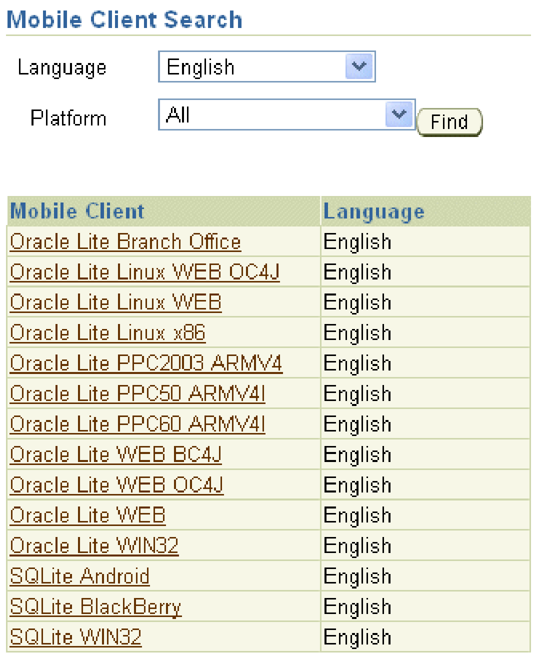
   Figure 2–1 displays the Mobile client setup page, which contains links to install Mobile client software for multiple platforms and languages. You can select another language than English on the Language pulldown.

   For viewing platforms, you can choose to see all available platforms for the indicated language, or only those platforms for Windows or WinCE with the Platform pull-down menu.

   Client platforms are provided in the Mobile client setup page for the Windows and WinCE platform: Pocket PC 2003 (PPC2003), Pocket PC SDK 5.0 (PPC50), and Windows Mobile 6 Professional SDK (PPC60). In addition, these client CAB files are optimized for size to minimize the footprint on your device.

If you are using a client with the Standard SDK for WinCE 5.0 platforms for Windows Mobile 5 (WCESTDSDK), then use the appropriate CAB files that are provided in the MDK install. For information on how to install the WCESTDSDK CAB files, see Section 2.3.3.1, "Installing Standard SDK WinCE 5.0 CAB Files for Your Mobile Client".

Figure 2–1 Mobile Client Setup Page



Language: English

Platform: All

Mobile Client	Language
Oracle Lite Branch Office	English
Oracle Lite Linux WEB OC4J	English
Oracle Lite Linux WEB	English
Oracle Lite Linux x86	English
Oracle Lite PPC2003 ARMV4	English
Oracle Lite PPC50 ARMV4I	English
Oracle Lite PPC60 ARMV4I	English
Oracle Lite WEB BC4J	English
Oracle Lite WEB OC4J	English
Oracle Lite WEB	English
Oracle Lite WIN32	English
SQLite Android	English
SQLite BlackBerry	English
SQLite WIN32	English

Note: Available clients may differ from what is shown above.

2. Click the SQLite Mobile client for your language and client platform.
3. The Save As dialog box appears. The file name field displays the setup executable file for the selected platform as an `.exe` file type. Save the executable file to a directory on the client machine.

Note: For WinCE, install any of the Oracle Database Lite Windows Mobile platforms to ActiveSync. Then, when the device is put into the cradle, ActiveSync installs the Oracle Database Lite on the device when it synchronizes.

4. Install the Mobile client. For all platforms, except installing WinCE on ActiveSync, go to the directory where you saved the setup executable file. Double-click the file to execute it.
5. Enter the username and password for the Mobile user.

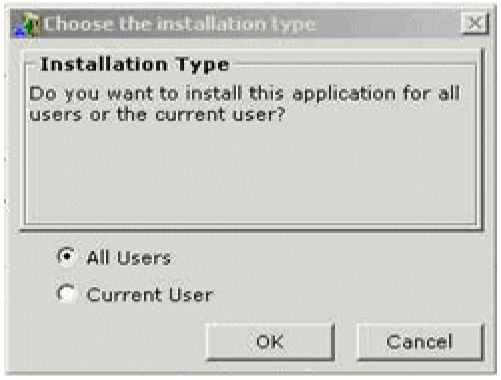
Note: For the restrictions on creating the username and password, see Section 5.3.1.2.1, "Defining Username and Password" in the *Oracle Database Lite Administration and Deployment Guide*.

6. You may be required to select the type of privilege under which to install the Mobile client. This may already be designated by the administrator in the INF file before installation or the current user may have a privilege that defaults to a certain privilege for the installation.

   - All Users—The user installing this Mobile client has administrator privileges and can install the Mobile client.
   - Current User—Selecting this option designates that the user does not have administrator privileges, but can install and use the Mobile client as a single user.

   Note: For details on how to designate the user privilege and for more information on user installation types, see Section 7.1, "Customize the Mobile Client Software Installation for Your Mobile Device" in the *Oracle Database Lite Administration and Deployment Guide*.

Figure 2–2 Select Installation Privileges


Choose the installation type

Installation Type

Do you want to install this application for all users or the current user?

[✓ All Users] Current User

7. Provide the client directory name where to install the Mobile client.
8. Once installed, synchronize the Mobile client for the first time. During the first synchronization, all applications and data for this user is brought down and installed on your Mobile client.
9. Each platform has further steps. See Table 2–6 for a description of the steps for each platform.

   Note: See Section 2.5, "Configuring for Automatic Synchronization When Installing the Client" for directions on how to enable a default synchronization after any client installation on your device.

Table 2–6 Initializing the First Synchronization for Each Mobile Client Platform

SQLite Mobile Client	Initial Synchronization Details
SQLite PocketPC for WinCE devices	Perform the following steps. **1.** If you install the PocketPC platform to ActiveSync, insert the WinCE device in the cradle. ActiveSync performs a synchronization to install Oracle Database Lite on the device. **2.** After SQLite Mobile client is installed on the device, then start the Device Manager Agent on the device by either selecting Device Manager in the programs group or by executing `dmagent.exe`, which is in the `orace` directory. **3.** Enter the username, password and Mobile Server URL. You can either enter the complete URL of the Mobile Server, the IP address or the hostname of the Mobile Server. If left off, the prefix "`http://`" is added automatically. Only use the hostname if the device is properly configured to use DNS name resolution. Otherwise, enter the IP address. The device is now registered with the Mobile Server and ready to be used.
All other platforms	Perform the following steps. **1.** Locate the directories where you installed the runtime libraries, and launch the Mobile Sync application. **2.** The `mSync` dialog appears. Enter the user name and password of the Mobile user. If you do not know your user name and password, ask your system administrator, who creates users and assigns passwords to each user. In the **Server** field, enter the URL for your Mobile Server. Click **Apply** and click **Sync**.

2.3.3.1 Installing Standard SDK WinCE 5.0 CAB Files for Your Mobile Client

By default, the Windows 2003, Windows Mobile 5 and Mobile 6 CAB files are installed with the Mobile Server and thus, are displayed as options on the Mobile client setup page. These CAB files are registered with the Mobile Server. However, if your Mobile client is a Standard SDK WinCE 5.0 platform, use one of the WCESTDSDK CAB files contained in the MDK install.

A SQLite Mobile client platform consists of a CAB file, an Installation Configuration File (INF file) that describes how to install the files, and an INI file that specifies the platform.

The following steps describe how to install the Standard SDK WinCE platform:

1. The WCESTDSDK CAB file must be copied from the MDK install to either the Mobile Server or the Mobile client, as described below:

   - Option One: Register the WCESTDSDK CAB file on the Mobile Server. The Mobile Server client setup page displays the predefined client platforms that you can download and install on your Mobile client device.

     If you want the Standard SDK WinCE CAB files to be displayed on the Mobile Server client setup page, then register the desired platform in the Mobile Server. After registration, the Mobile client can download the SDK CAB file from the Client setup page.

     Find the unregistered CAB file for the desired platform and language in the MDK installation in the following directory:

     ```
     <ORACLE_HOME>\Mobile\SDK\wince\<platform>\cabfiles
     ```

Copy and rename the CAB file. The SQLite CAB files are named `sqlite_sync.<language>.<platform>.<cpu>.cab`. Rename the CAB file to `sqlite_sync.cab` and copy it into a subdirectories according to language and client platform type relative to the `<ORACLE_HOME>\mobile_oc4j\j2ee\mobileserver\applications\mobileserver\setup\` directory. Take note of the directory path as you will provide the location of the CAB file in the INF file. in the INF file.

- Option Two: Copy the desired WCESTDSDK CAB file directly to the Mobile client from the `<ORACLE_HOME>\Mobile\SDK\wince\<platform>\cabfiles\` directory. Each CAB file is named `sqlite_sync.<language>.<platform>.<cpu>.cab`.

2. On the Mobile Server, create an INF file and place it in the appropriate subdirectory according to the language and platform type in the `ORACLE_HOME\mobile_oc4j\j2ee\mobileserver\applications\mobileserver\setup\dmc` directory. The INF file provides the instructions for installing the CAB file on the client platform. You can copy one of the existing INF files, such as the `std500.inf` file. If you want to add additional instructions, copy the file and make sure the INI file refers to the new INF file.

   If you have to modify it for the new platform, make sure that you give it a new name to avoid changing an existing platform. Provide the location of the CAB file—which you found in step 1—in the `<file><item><src>` and `<des>` tags, which are described in Section 7.10, "Installation Configuration (INF) File"in the *Oracle Database Lite Administration and Deployment Guide*.

   The following demonstrates how to specify a CAB file located in the `WINCE/<language>/stdsdk500/<cpu>` directory, which is relative to the `setup` directory, and the destination for the CAB file.

```
<file>
  <item type='WINCE'>
    <src>/$OS_LANG$/stdsdk500/$CPU$/sqlite_sync.cab</src>
    <des>$APP_DIR$\sqlite_sync.cab</des>
  </item>
</file>
```

3. On the Mobile Server, create an INI file that refers to the INF file for this platform. See Section 2.3.3.1.1, "Defining the INI File" for details.
4. On the Mobile Server, register the new platform with the device manager resource loader, which uses the INI script to create a new Platform.

```
ORACLE_HOME\mobile\server\admin\dmloader
        <repository_owner>/<repository_password>@jdbc_url <ini_filename>
```

   For example, to load the `std500.ini` file as shown in step 3, perform the following:

```
ORACLE_HOME\mobile\server\admin\dmloader
        <repository_owner>/<repository_password>@jdbc_url std500.ini
```

Note: if you supply a RAC URL as the JDBC URL, then enclose it within two double-quotes as the operating system treats the equal sign (=) as a delimiter, which truncates the RAC URL and throws the `syntax error: unexpected token '('.` error

5. On the Mobile Server, copy the `setup_<language>.exe` files to the following directory on the Mobile Server:

   ```
   <ORACLE_HOME>\mobile_oc4j\j2ee\mobileserver\applications\mobileserver
   \setup\dmc\wince\<platform>\<chipset>\
   ```

   For example, registering the `wcestd500_sdk` CAB file, the setup files should be copied to the following directory:

   ```
   ORACLE_HOME\mobile_oc4j\j2ee\mobileserver\applications\mobileserver
   \setup\dmc\wince\ppcstd500\armv4i
   ```

6. Restart the Mobile Server to see the newly registered platform in the setup GUI.
7. On the client, open a new browser that points to the setup page to select the newly registered platform with the SDK CAB file.

2.3.3.1.1 Defining the INI File Create an INI file that refers to the INF file, as well as other attributes. The following shows how the INI file is organized:

```
# List platforms to be created in the [Platform] section
#
# Format: platform_name;language
[PLATFORM]
# Provide string to be displayed in the setup UI
PLATFORM1;LANGUAGE
#
# Platform details. One entry for each platform listed in the
#[PLATFORM] Section. Provide the same info but prepend with "PLATFORM."
[PLATFORM.PLATFORM1;LANGUAGE]
TYPE=OS_CPU_LANGUAGE_NAME
INF=file.inf
BOOTSTRAP=dmcommand
ATTRIBUTES=attribute1=value1&attribute2=value2
```

Where the tags define the following:

- `PLATFORM`: Provide the platform type and language separated by a semi-colon.
- `TYPE`: Provide a name for the platform that is a concatenation of the operating system, CPU, language, and name—where each are separated by an underscore—such as `WINCE_ARMV4I_US_OLITE_STD500`.
- `INF`: Provide the name of the INF file, such as `sqlite_win32.inf` or `sqlite_linux.inf`.
- `BOOTSTRAP`: You can find a list of the bootstrap commands in a pull-down in the Mobile Devices page.
- `ATTRIBUTES`: The attributes are separated by an ampersand (&). These are the same attributes that are discussed in Section 7.4.3.2, "Create a Custom Platform By Extending an Existing Platform" in the *Oracle Database Lite Administration and Deployment Guide* and are as follows:
  - Can the device be updated: `update=true|false`
  - Is the platform enabled: `enabled=true|false`
  - Can applications on the device be updated: `app_upgrade=true|false`
  - Should the device manager on the client be started automatically: `dmc=auto`

For example, the following is an INI file that describes the WinCE Standard SDK 5.00 for ARMV4I:

```
# Platforms
#
[PLATFORM]
# Windows CE Standard SDK 5.00 - ARMV4I
# Provide string to be displayed in the setup UI
SQLite WCESTD500 ARMV4I;US
#
# Windows CE Standard SDK 5.00 ARM V4i
[PLATFORM.SQLite WCESTD500 ARMV4I;US]
TYPE=WINCE_ARMV4I_US_OLITE_STD500
INF=std500.inf
BOOTSTRAP=DeviceInfo
ATTRIBUTES=dmc=auto&update=true&enabled=true
```

2.4 Configuring the Location of the SQLite Database and Mobile Client Files

The location of the SQLite database is determined by the `SQLITE.DATA_DIRECTORY` parameter in the `OSE.INI` file.

- All SQLite databases and temporary synchronization data are stored in the `SQLITE.DATA_DIRECTORY/sqlite_db/`*`<user>`* directory, where *`<user>`* is the synchronization user id. These are named with the `.db` extension, such as `TERRY\mysqlite.db`. These files are used to manage the change control for transactions and synchronization for the user.
- Internal settings and parameters for the SQLite Mobile client is stored in the `SQLITE.DATA_DIRECTORY/oseconf` directory.

The following shows an example of configuring the SQLite database directory on a Win32 platform:

```
SQLITE.DATA_DIRECTORY=C:\mobileclient\sqlite
```

For more details on this parameter, see Appendix A.1.1, "SQLITE.DATA_DIRECTORY".

2.5 Configuring for Automatic Synchronization When Installing the Client

In the default configuration, Mobile clients do not automatically synchronize after you install the client. However, for Win32, WinCE, Windows Mobile or Linux platforms, you can modify your configuration to automatically synchronize each client after it is installed, as follows:

1. Logon to the Mobile Server as an Administrator and launch the Mobile Manager tool.
2. Click on Mobile Devices, followed by Administration.
3. Click on Command Management.
4. Edit the Command Device Info (Retrieve device information).
5. Insert 'Synchronize' as a Selected Command and click **Apply** to accept the changes.

See Section 7.6, "Sending Commands to Your Mobile Devices" in the *Oracle Database Lite Administration and Deployment Guide* for more details on sending commands to your Mobile device.

3 Managing Your SQLite Mobile Client

The following sections describe how to manage the Oracle Database Lite functionality on the SQLite Mobile client:

- Section 3.1, "Starting the SQLite Mobile Client"
- Section 3.2, "Synchronize Data for Applications on the SQLite Mobile Client"
- Section 3.3, "Use the mSync GUI to Initiate Synchronization"
- Section 3.4, "Reset the Mobile User Password"
- Section 3.5, "Manage Snapshots on SQLite Mobile Client"
- Section 3.6, "Control Automatic Synchronization for a Specific SQLite Mobile Client"
- Section 3.7, "Improve Performance by Disabling the Resume Feature"
- Section 3.8, "Use the Device Manager Client GUI to Manage the Client-Side Device"
- Section 3.9, "Initiate Updates for the SQLite Mobile Client"
- Section 3.10, "Communicate Between the Internet and Intranet Through a Reverse Proxy"

3.1 Starting the SQLite Mobile Client

When you installed the SQLite Mobile client on Linux or Windows, it configured that the Mobile client should always be started automatically when the device is initiated. For Win32 and WinCE devices, you do not have to perform anything extra to start the Mobile client. However, for the Linux platform, you may have to start the Linux Mobile client by executing the `webtogo.sh` file in the `<mobile_client>/bin`.

3.2 Synchronize Data for Applications on the SQLite Mobile Client

You can have an application downloaded onto a device, where data can be synchronized between the SQLite Mobile client and the back-end Oracle database.

The following describes how to initiate synchronization from each type of SQLite Mobile client:

- Blackberry and Android clients: The application built for these clients initiate synchronization by executing the SQLite Mobile client Java APIs. For details on synchronization APIs, see Chapter 2, "Synchronization" and Chapter 3, "Invoking Synchronization APIs from Applications" in the *Oracle Database Lite Developer's Guide* for more information. For full details on the Java APIs, see the Javadoc

- Linux, Win32, and WinCE clients: The application built for these clients can use the SQLite Mobile client Java APIs or C/C++ APIs. Thus, start the application as you would start any application on these platforms.

Note: When you initiate a synchronization from the client, either manually or by scheduling a job, the synchronization cannot occur if there is an active connection with an uncommitted transaction opened from another source. This could be from scheduling two jobs to synchronize at the same time, from mSync, or the client synchronization APIs.

Initiate synchronization through one of the following methods:

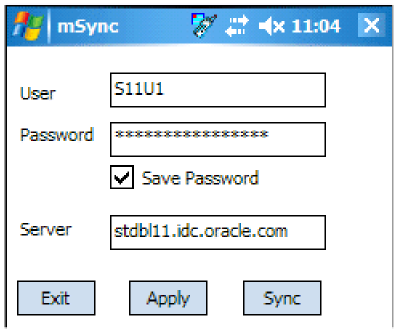
- Execute the `msync` executable, described in Section 3.3, "Use the mSync GUI to Initiate Synchronization".
- Implement synchronization within your application using the synchronization APIs, as described in Chapter 2, "Synchronization" and Chapter 3, "Invoking Synchronization APIs from Applications" in the *Oracle Database Lite Developer's Guide*.

Note: The Mobile client device clock must be accurate for the time zone set on the device before attempting to synchronize. An inaccurate time may result in the following exception during synchronization: `CNS: 9026 "Wrong username or password. Please enter correct value and reSync."`

3.3 Use the mSync GUI to Initiate Synchronization

You can initiate synchronization of the SQLite Mobile client using the mSync GUI, as shown in Figure 3–1.

Figure 3–1 Using the mSync GUI to Initiate Synchronization



User S11U1
Password ****************
[✓] Save Password
Server stdbl11.idc.oracle.com

To bring up the mSync GUI, execute `msync.exe` on WinCE and Win32 or `msync` on Linux, which is located in the `/SQLite` subdirectory under the directory where you installed the Mobile client. For Blackberry and Android platforms, start mSync by clicking the mSync application icon.

Modify the following supplied values, if incorrect:

- Username and password for the user that is starting the synchronization.

Note: See Section 4.3.1.2.1, "Define Username and Password" in the *Oracle Database Lite Administration and Deployment Guide* for conventions for creating the username or password.

- Check if you want the password saved for future requests.
- Host name where the Mobile Server is installed.

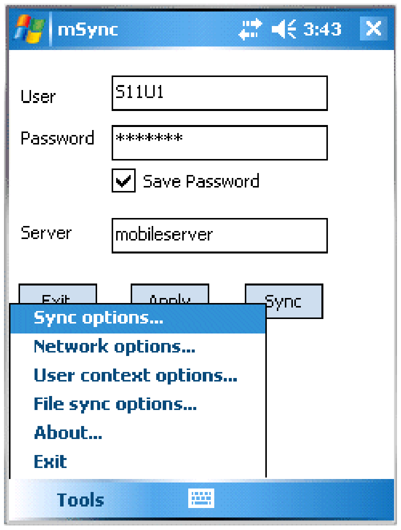
Click **Sync** to start the Synchronization. Click **Apply** to save any modifications you made to the entries. Click **Exit** to leave the tool.

If there are software updates that are waiting to be downloaded to the client, then the update tool is automatically executed after the end of the synchronization process. See Section 3.9, "Initiate Updates for the SQLite Mobile Client" for more information.

Note: The only time that the client does not check for software updates is if you are using the Synchronization APIs. If you want to launch the update UI, then enter `update` on the command line.

You can also modify the tool options by selecting the Tools Selection at the bottom of the UI, as shown in Figure 3–2.

Figure 3–2 The mSync Tools Selection



User: S11U1
Password: ********
[✓] Save Password
Server: mobileserver

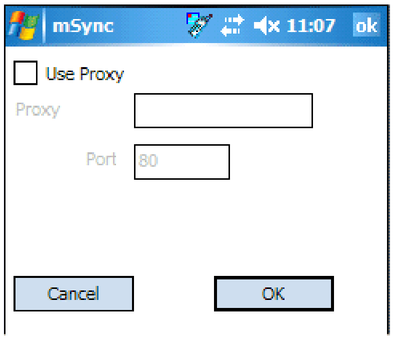
The following sections describe the Tools options:

- Section 3.3.1, "Network Options for MSync Tool"
- Section 3.3.2, "Sync Options for MSync Tool"
- Section 3.3.3, "Sync to a File Using File-Based Sync"
- Section 3.3.4, "Use SQLite Mobile Client Tools on Linux"

3.3.1 Network Options for MSync Tool

Figure 3–3 displays the Network options screen where you can specify a proxy if your network provider requires that you use a proxy server to access the internet. Click **Use Proxy** to use a proxy and then enter the proxy server and port number.

Figure 3–3 The mSync Network Options Selection



3.3.2 Sync Options for MSync Tool

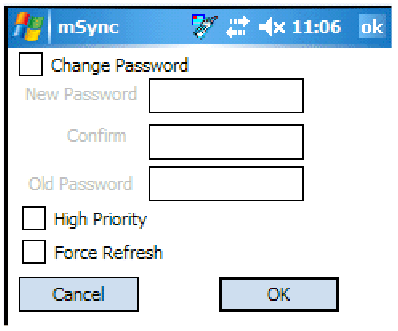
Figure 3–4 displays the Sync Options screen where you can specify the following:

- Mobile User Password—Modify the existing password. The Mobile user password is stored on both the client and the Mobile Server. To ensure that both are modified, only change the password when connected to the Mobile Server. See Section 3.4, "Reset the Mobile User Password" for details.
- High Priority—Select this checkbox to specify synchronizing only High Priority data. This specifies under what conditions the different priority records are synchronized. By default, the value is LOW, which is synchronized last. If you have a very low network bandwidth and a high ping delay, you may only want to synchronize your HIGH priority data.

  When you select this checkbox, you are enabling pre-defined high priority records to be synchronized first. This only for those publication items that have specified a restricting predicate. See Section 1.2.10, "Priority-Based Replication" in the *Oracle Database Lite Troubleshooting and Tuning Guide* for more information.
- Force Refresh—The force refresh option is an emergency only synchronization option. Check this option when a client is corrupt or malfunctioning, so that you decide to replace the Mobile client data with a fresh copy of data from the enterprise data store with the forced refresh. When this option is selected, any data transactions that have been made on the client are lost.

  When a force refresh is initiated all data on the client is removed. The client then brings down an accurate copy of the client data from the enterprise database to start fresh with exactly what is currently stored in the enterprise data store.

Figure 3–4 The mSync Options Selection



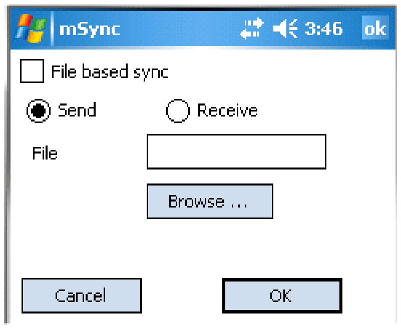
3.3.3 Sync to a File Using File-Based Sync

Once you select File Based Sync off the Tools menu, the screen shown in Figure 3–5 is displayed. To synchronize to a file, click on the File based sync checkbox and perform the following:

- If you select the send radio button, then browse for a directory where you want the client to save the upload data file from the Mobile client for the Mobile Server.
- If you select the receive radio button, then provide the location for the download data file from the Mobile Server.

For full details on File-Based Sync, see Section 5.8, "Synchronizing to a File with File-Based Sync" in the *Oracle Database Lite Administration and Deployment Guide*.

Figure 3–5 File Sync Options



3.3.4 Use SQLite Mobile Client Tools on Linux

The SQLite Mobile client for Linux supports the `msync`, `dmagent` and `update` tools. To use the UI-based tools, use the following executables: `msync`, `dmagent`, or `update`.

To synchronize on a Linux client with the command line tool, use the `msync` executable for synchronization, as follows:

```
./msync username/password@server[:port][@proxy:port]
```

For example,

```
./msync john/john@testserver:8000
```

The other `msync` options, such as `-save`, `-a`, `-password` and `-force` currently will not result in a successful sync. This is a limitation only for the `msync` executable in the MDK installation on Linux.

3.4 Reset the Mobile User Password

Because the Mobile user password is stored on both the client and the Mobile Server, modify the password as follows:

- Modify the password on the client using the mSync UI. Only modify the password with the mSync UI if you are connected to the Mobile Server to ensure that the user password change is propagated to the Mobile repository.
- Modify the Mobile user password in the Mobile Manager in the User Properties page. If you simply want to invalidate the Mobile user, then you only have to modify the password on this screen; however, if you want to reset the password

on both the Mobile Server and the Mobile user, then also send a Reset Password command from the Device Management section in the Mobile Manager to the Mobile client.

After sending the Reset Password command, you need to perform a synchronization on the client with the new password. Then, you will be able to connect to the client database using the new password.

Note: If you modify the password on the server and do not send the Reset Password, then the client cannot synchronize. In this case, either send the Reset Password or return the password back to its original value on the server before retrying the synchronization.

See Section 11.2, "Which Password is Which" in the *Oracle Database Lite Administration and Deployment Guide* for details on passwords.

3.5 Manage Snapshots on SQLite Mobile Client

The following are the types of snapshots you can enable for tracking the changes on the SQLite database:

- *State-based*. State-based snapshots decipher the difference in the state of the data between subsequent synchronization events. This snapshot type is more resource efficient than queue-based snapshots. The SQLite Mobile client Java APIs only support state-based snapshots. To enable state-based snapshots, set the `SQLITE.QUEUES` parameter in the `OSE.INI` file to `NO`.

  Snapshot state tables, `OSE_ST$<snapshot>`, are created in the SQLite database and are populated by SQL triggers with primary keys of the modified rows.

- *Queue-based*: Both client and server changes are stored in a single queue. Whenever the snapshot is not locked by an application, the synchronization retrieves data from the In Queue and applies it to the base snapshot. At this point, the synchronization propagates data from the Out Queue to the server.

  Although both snapshot types rely on triggers, queue-based snapshots allow concurrent operations on the SQLite database while any synchronization is in progress. The Sync Agent compose operation places modified data into the Out Queue. Later, the sync session uploads multiple client transactions delineated by a unique transaction id to the server.

  To enable queue-based snapshots, set the `SQLITE.QUEUES` parameter in the `OSE.INI` file to `YES`. This is the default.

  When you use queue-based snapshots, a queue database file is created, which is named `OSE_<database name>.db`. This database file contains the following tables:

  - Data queue for both In Queue and Out Queue records named `OSE$DATAQ`.
  - BLOB queue named `OSE$BLOBQ`.
  - Snapshot registry named `OSE$TABLES`.
  - Transactions registry named `OSE$TRANS`.
  - Transaction sequences per publication in the `OSE$TRSEQ` table,

  The `OSE$DATAQ` queue is used for all snapshots and contains both In and Out Queue records. The TRID column is positive when the record is an Out Queue record. When you synchronize with queue-based snapshots enabled, new data from the client is uploaded from the `OSE$DATAQ` queue table and new data from the Oracle database is downloaded into this queue.

For more details on this parameter, see Appendix A.1.2, "SQLITE.QUEUES".

3.6 Control Automatic Synchronization for a Specific SQLite Mobile Client

As described in Section 5.4, "Using Automatic Synchronization" in the *Oracle Database Lite Administration and Deployment Guide*, you can enable automatic synchronization for native Mobile clients either in the publication item or for the entire platform.

However, you can disable automatic synchronization for a single client by configuring the `BGSYNC.DISABLE` parameter to `YES` in the `OSE.INI` file on the SQLite Mobile client. This disables the Sync Agent and the only method for synchronization is a manual synchronization.

For more details on this parameter, see Appendix A.1.4, "BGSYNC.DISABLE".

3.7 Improve Performance by Disabling the Resume Feature

The resume feature manages intermittent network failures. If resume is enabled on both the server and the client, synchronization will resume automatically within the specified resume timeout period. Also, if sync session was interrupted during a network operation, the next synchronization will try to resume the operation, as long as resume is enabled and the resume timeout has not expired.

The resume transport adds overhead with additional network round trips and additional data to be saved on the client and on the server. Any device with reliable networks may disable the resume feature to improve performance of the synchronization system for this device and improve scalability on the server.

You can disable the resume feature for the SQLite Mobile client device by setting the `OSE.RESUME` parameter in the `OSE.INI` file to NO. For more details on the resume feature and disabling it for your SQLite Mobile client, see Section A.1.3, "OSE.RESUME" and Section 5.6, "Resuming an Interrupted Synchronization" in the *Oracle Database Lite Administration and Deployment Guide*.

3.8 Use the Device Manager Client GUI to Manage the Client-Side Device

On Win32, WinCE, or Linux client platforms, you can manage the client software using the Device Manager. See Section 7.8, "Using the Device Manager Agent (dmagent) on the Client" in the *Oracle Database Lite Administration and Deployment Guide* for a full description.

3.9 Initiate Updates for the SQLite Mobile Client

You can initiate a request for software updates from the Mobile Server by executing the Oracle Database Lite Update tool. For details, see Section 7.7.3, "Initiate Updates of Oracle Database Lite Software for Mobile Clients" in the *Oracle Database Lite Administration and Deployment Guide*.

3.10 Communicate Between the Internet and Intranet Through a Reverse Proxy

If a Win32, WinCE or Linux SQLite Mobile client is on either side of the firewall, you set up a proxy or reverse proxy to facilitate communication between the Mobile client and Mobile Server. See Section 11.6, "Using a Firewall Proxy or Reverse Proxy" in the *Oracle Database Lite Administration and Deployment Guide*.

4

Using an Android Application on the SQLite Mobile Client

As described on the Android developer Web site at `http://developer.android.com`, all applications for the Android platform are required to be digitally signed in order to be installed on the Android device. This key is also used to encrypt the SQLite database. Thus, only the user with the key can access the database and perform synchronization.

Because of the required key, you cannot download and install the SQLite Mobile client binaries unless they are downloaded within the context of a signed application. Thus, there is no option for a manual synchronization through mSync. Instead, all manual synchronization events are invoked through synchronization APIs.

Note: This chapter assumes that you know how to use Eclipse to build an Android project and how to appropriately develop and sign an Android application.

The following sections uses the `simple_sync_android` project to describe the steps to include the SQLite Mobile client within your signed application.

- Section 4.1, "Prerequisites"
- Section 4.2, "Developing an Android Application for Oracle Database Lite"
- Section 4.3, "Import the Oracle Database Lite Android Project into Eclipse"
- Section 4.4, "Build Oracle Database Lite Android Project"

4.1 Prerequisites

The following are the prerequisites for enabling synchronization for a SQLite application:

1. Install Eclipse IDE with the ADT plug-in, as detailed at the following site:

   `http://developer.android.com/sdk/eclipse-adt.html#installing`

2. Install the latest Android SDK, as detailed at the following site:

   `http://developer.android.com/sdk/index.html`

3. Install the Oracle Database Lite Mobile Development Kit.

4.2 Developing an Android Application for Oracle Database Lite

When developing the application for an Android platform, perform the following to include the SQLite Mobile client:

1. Use the Oracle Database Lite pure Java synchronization APIs to invoke a manual synchronization. For more information on the pure Java APIs, see Section 4.3, "Synchronization API for Pure Java Applications on SQLite Mobile Clients" in the *Oracle Database Lite Developer's Guide*.

2. Sign and build your application according to Android specifications.

An Android platform sample named `MainAct.java` is included in the `<MDK_ROOT>\Mobile\Sdk\samples\Sync\android\simple_sync_android` directory. The `MainAct.java` sample initializes the required synchronization structures and invokes the synchronization APIs.

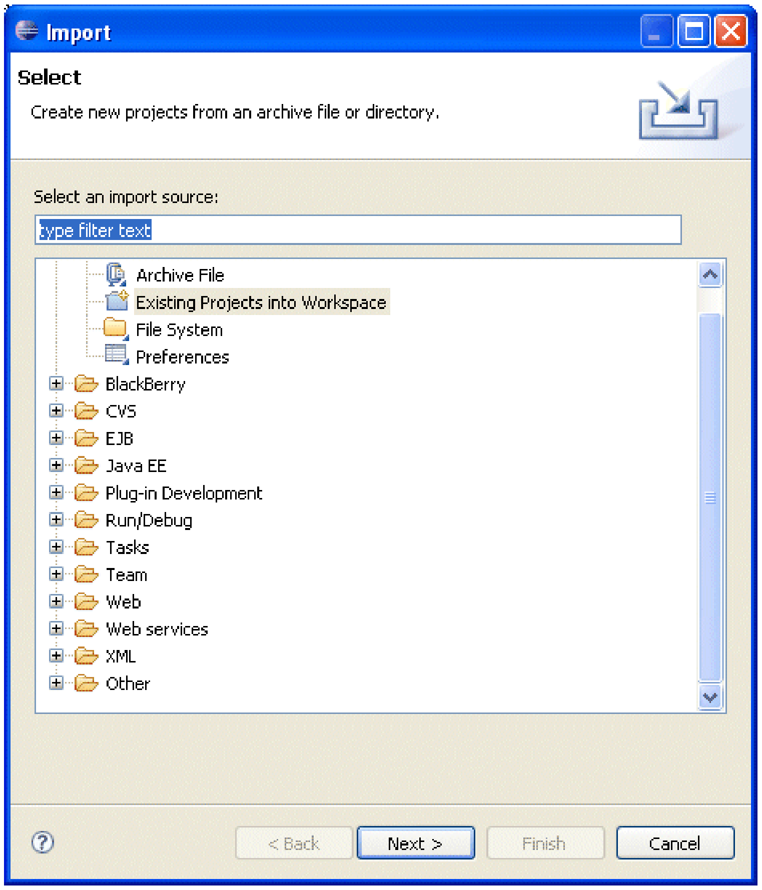
4.3 Import the Oracle Database Lite Android Project into Eclipse

The following steps show how to import your Android project using the Oracle Database Lite `simple_sync_android` sample project.

1. Import the Oracle Database Lite sample Android project into your Eclipse Workspace.

   a. Select File->Import and then choose **Existing Projects into Workspace**. Click **Next**.

Figure 4–1 Import Existing Projects into Eclipse Workspace



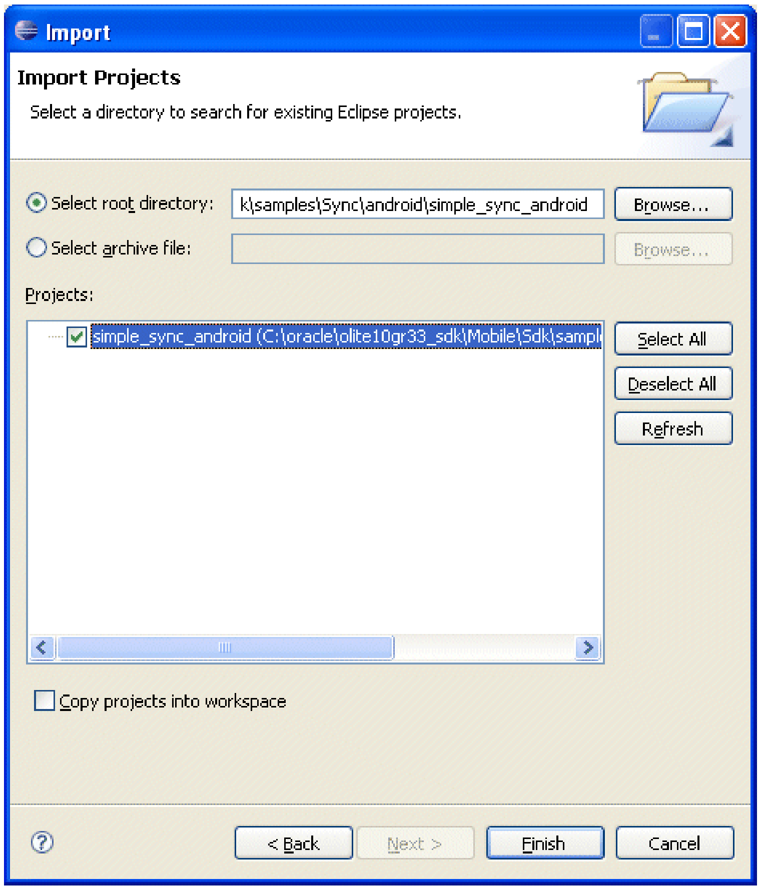
b. Set the root directory to point to the Android project within the Oracle Database Lite MDK. Enable the **Select Root Directory** button and browse for the `simple_sync_android` project, which is located in the following directory:

```
<MDK_ROOT>Mobile\Sdk\samples\Sync\android\simple_sync_android
```

where the *<MDK_ROOT>* is replaced with the full path where the Oracle Database Lite MDK is installed.

Figure 4–2 demonstrates setting the root directory. After which, all projects in the specified root directory are displayed in the Projects window.

Figure 4–2 Select Root Directory for Eclipse Project



Select root directory: k\samples\Sync\android\simple_sync_android

Select archive file:

Projects:

☑ simple_sync_android (C:\oracle\olite10gr33_sdk\Mobile\Sdk\sampl

☐ Copy projects into workspace

c. Select the `simple_sync_android` project and click **Finish**. The `simple_sync_android` project is now imported into your Eclipse Workspace.

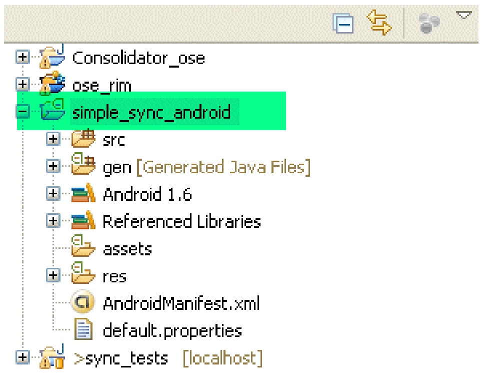
4.4 Build Oracle Database Lite Android Project

The following details how to build your Android project using the Oracle Database Lite `simple_sync_android` sample project.

1. Set required environment variables. The project references Oracle Database Lite synchronization classes, which are located within the `osync_android.jar` library file. Set the `MOBILE_SYNC_ANDROID_LIB` environment variable to point to the location of this JAR file with the following steps:

a. Highlight the `simple_sync_android` project in the Project Explorer window, as shown in Figure 4–3.

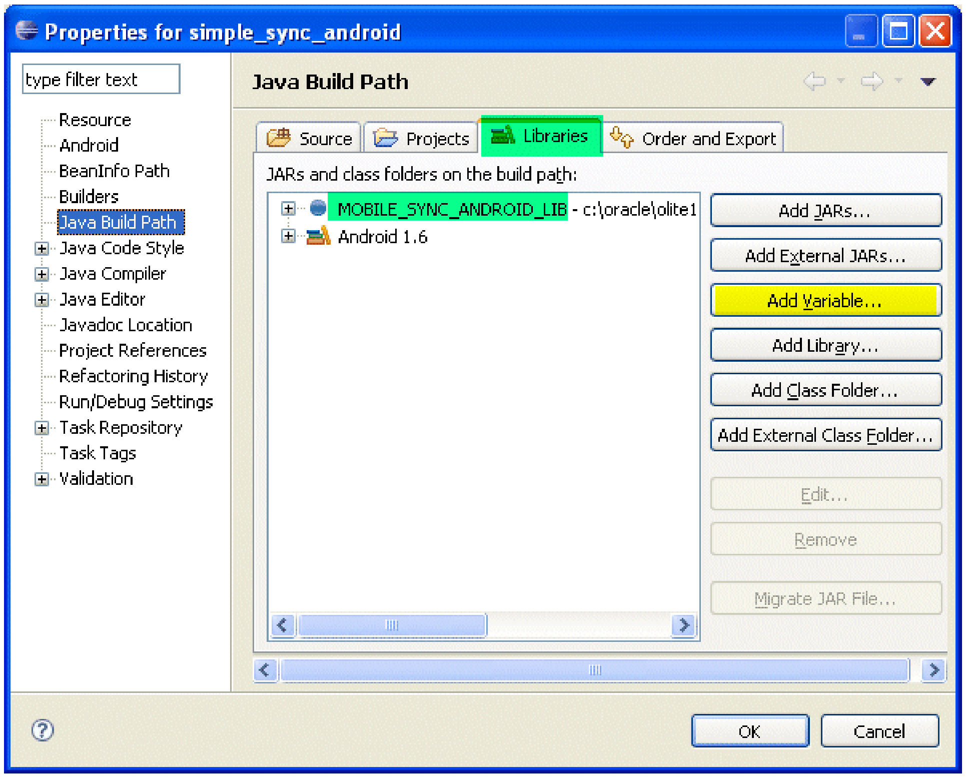
Figure 4–3 Project Explorer window



b. Select '**Alt Enter**' to display the Project Properties window.

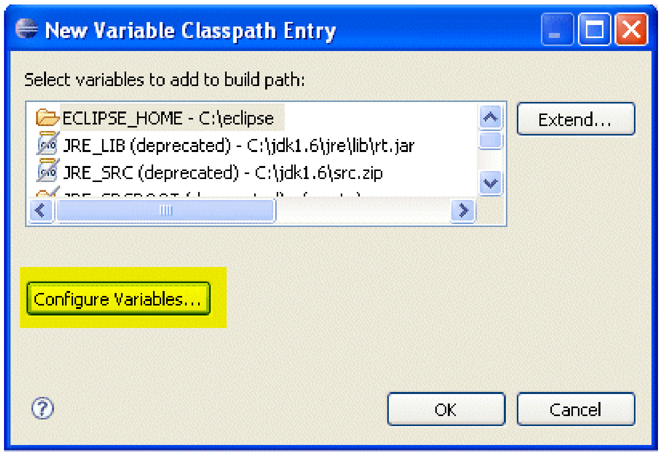
c. As shown in Figure 4–4, select the **Java Build Path** in the left pane. Then, select the **Libraries** tab in the Java Build Path window. Click **Add variable**.

Figure 4–4 Properties for simple_sync_android Project



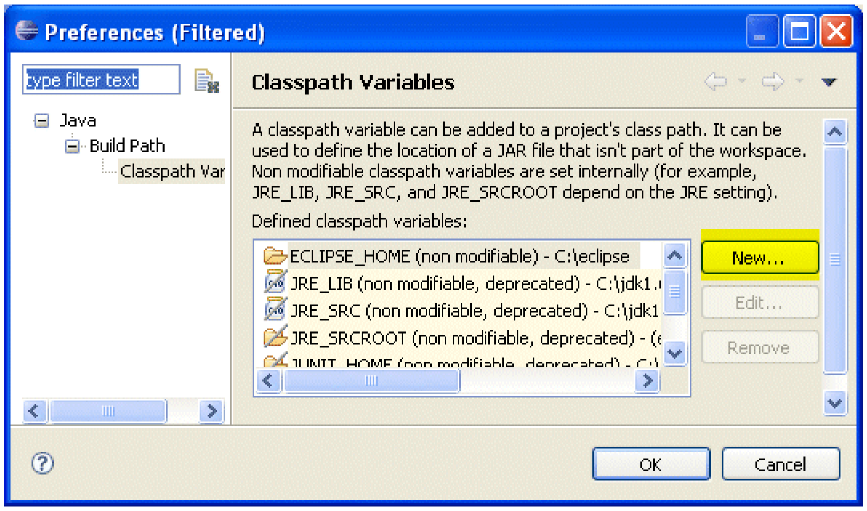
d. Click **Configure Variables**.

Figure 4–5 Configure a New Variable


New Variable Classpath Entry
Select variables to add to build path:
ECLIPSE_HOME - C:\eclipse
JRE_LIB (deprecated) - C:\jdk1.6\jre\lib\rt.jar
JRE_SRC (deprecated) - C:\jdk1.6\src.zip
Extend...
Configure Variables...
OK
Cancel

e. Click **New** to add a the new variable.

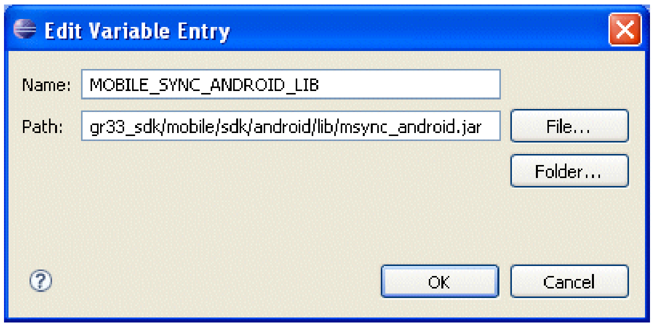
Figure 4–6 Add New Path Variable


Preferences (Filtered)
type filter text
Java
Build Path
Classpath Var
Classpath Variables
Defined classpath variables:
ECLIPSE_HOME (non modifiable) - C:\eclipse
JRE_LIB (non modifiable, deprecated) - C:\jdk1.
JRE_SRC (non modifiable, deprecated) - C:\jdk1
JRE_SRCROOT (non modifiable, deprecated) - (
New...
Edit...
Remove
OK
Cancel

A classpath variable can be added to a project's class path. It can be used to define the location of a JAR file that isn't part of the workspace. Non modifiable classpath variables are set internally (for example, JRE_LIB, JRE_SRC, and JRE_SRCROOT depend on the JRE setting).

f. Enter `MOBILE_SYNC_ANDROID_LIB` in the Name field and add *<MDK_ROOT>*`/Mobile/Sdk/android/lib/osync_android.jar` in the Path field.

Figure 4–7 Creating the MOBILE_SYNC_ANDROID_LIB Environment Variable



Name: MOBILE_SYNC_ANDROID_LIB
Path: gr33_sdk/mobile/sdk/android/lib/msync_android.jar

g. When finished, click **OK**. The `MOBILE_SYNC_ANDROID_LIB` variable provides the library directory where the `osync_android.jar` file is located, which is required to build your project.

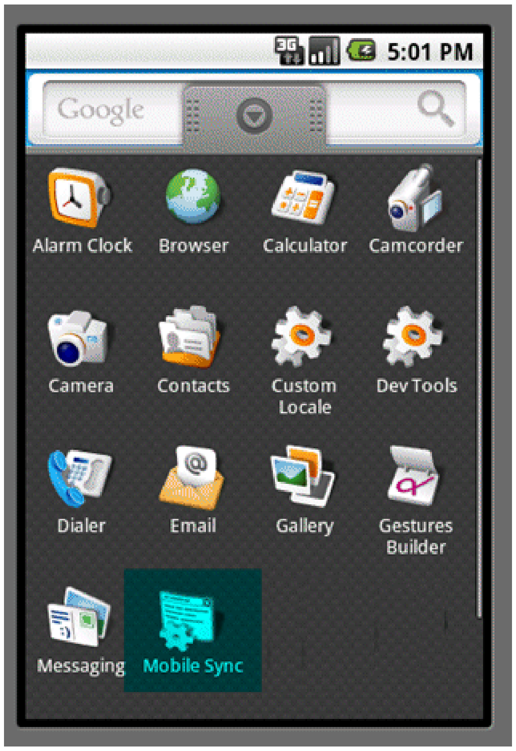
2. Refresh and build the `simple_sync_android` project to link in the `osync_android.jar` file with the Oracle Database Lite synchronization libraries.

3. Execute and debug the `simple_sync_android` project with the Android emulator.

   a. Right click on the `simple_sync_android` project.

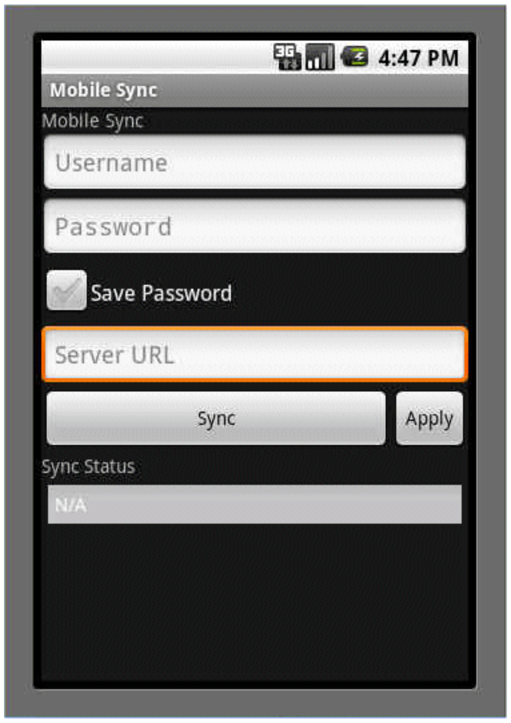
   b. Click **Run As** and select **Android Application**. The Android emulator is started, where you can execute the sample as the **Mobile Sync** application.

Figure 4–8 Test Project with Android Emulator



4. When you execute the Mobile Sync application, synchronization is initiated. Enter your username, password and Mobile Server URL in the Sync UI, as shown in Figure 4–9.

Figure 4–9 Synchronization UI



5. After the initial synchronization, verify that the SQLite database files are created in the following directory on your Android device:

   ```
   /data/data/<pkg_name>/app_oracle.sync/sqlite_db
   ```

   where `<pkg_name>` is the name of your package. For example, the `simple_sync_android` sample stores its SQLite database files in the following directory:

   ```
   /data/data/tests.sync/app_oracle.sync/sqlite_db
   ```

6. Once you have completed testing your project using an Android emulator, perform the following:

   a. Build and sign your project according to Android standards as described on `http://developer.android.com`. All applications on the Android platform are required to be digitally signed with a certificate whose private key is held by the application's developer. The application developer signs the Android application with his or her own signing key. This key is also used to encrypt the SQLite database.

   The Android platform tests the signer certificate's expiration date at install time.

   b. Download your application to all intended Android devices.

A INI Configuration Parameters

You can customize the SQLite Mobile client by modifying the parameter values defined in two configuration files: `OSE.INI` and `POLITE.INI`.

The installation automatically sets the parameters in both the `OSE.INI` and `POLITE.INI` files, but you can modify them to customize the product behavior. To modify these files, use an ASCII text editor. You must have write permissions on the directory where either file is located to be able to modify them.

Note: On the WinCE and Blackberry platforms, these files are named with the extension of `.TXT`, so that you can double-click on it to open the file.

The following describes the configuration files for the SQLite Mobile client:

- Section A.1, "OSE.INI File Overview"
- Section A.2, "POLITE.INI File Overview"

A.1 OSE.INI File Overview

The `OSE.INI` file contains parameters that define the location of the SQLite database and SQLite Mobile client files, and how to customize synchronization for the SQLite database. There is a single `OSE.INI` file for each mobile device for all users of that device. The latest modifications to parameters in the `OSE.INI` file take effect only during a manual synchronization or after restarting the Sync Agent for automatic synchronization.

Depending on the platform, it can be located in one of the following directories on the mobile device:

- On native clients, the `OSE.INI` file is located in `<`*`mobile_client_install_root`*`>\sqlite`. On WinCE, this file is named `OSE.TXT`.
- On Blackberry, the `OSE.TXT` file is located in `/store/user/home/oracle/sync`.
- On Android, the `OSE.INI` file is located in `/data/data/<`*`application_package`*`>/app_oracle.sync`.

  Applications import the `mSync.jar` library; thus, the `<`*`application_package`*`>` should be replaced with the user's application that invokes the `OSESession` APIs.

The following are the parameters for the `OSE.INI` file:

- Section A.1.1, "SQLITE.DATA_DIRECTORY"
- Section A.1.2, "SQLITE.QUEUES"
- Section A.1.3, "OSE.RESUME"
- Section A.1.4, "BGSYNC.DISABLE"

A.1.1 SQLITE.DATA_DIRECTORY

By default, the location of the SQLite database is determined by the `SQLITE.DATA_DIRECTORY` parameter in the `OSE.INI` file. However, if this parameter is not set, the location of the SQLite database on Win32, WinCE, or Linux platforms is determined by the location of the `ospSqlite` library.

- The SQLite database, Oracle Database Lite repository files, and temporary synchronization data are stored in the `SQLITE.DATA_DIRECTORY/sqlite_db/<user>` directory, where `<user>` is the synchronization user id. The database repository files are named with the `.db` extension, such as `TERRY\mysqlite.db`. These files are used to manage the change control for transactions and synchronization for the user.
- Internal settings and parameters for the SQLite Mobile client is stored in the `SQLITE.DATA_DIRECTORY/oseconf` directory.

Example

Example for setting the directory on a Win32 platform:

```
SQLITE.DATA_DIRECTORY=C:\mobileclient\sqlite
```

Example for setting the directory on a Blackberry:

```
SQLITE.DATA_DIRECTORY=file:///SDCard/databases/my_app
```

A.1.2 SQLITE.QUEUES

The `SQLITE.QUEUES` parameter specifies which type of snapshots the client will use in tracking the changes on the SQLite database. The following list the two snapshot types:

- *Queue-based*: Both client and server changes are stored in a single queue. Whenever the snapshot is not locked by an application, the synchronization retrieves data from the In Queue and applies it to the base snapshot. At this point, the synchronization propagates data from the Out Queue to the server.

  Although both snapshot types rely on triggers, queue-based snapshots allow concurrent operations on the SQLite database while any synchronization is in progress. The Sync Agent's compose operation places modified data into the Out Queue. Later, the Sync Session uploads multiple client transactions delineated by a unique transaction id to the server.

  Set this type with `SQLITE.QUEUES`=YES.
- *State-based*. State-based snapshots decipher the difference in the state of the data between subsequent synchronization events. This snapshot type is more resource efficient than queue-based snapshots. Pure Java clients only support state-based snapshots. To enable queue-based snapshots, set the `SQLITE.QUEUES` parameter in the `OSE.INI` file to `NO`.

Syntax

```
SQLITE.QUEUES=YES|NO
```

A.1.3 OSE.RESUME

The `OSE.RESUME` parameter specifies whether the resume transport is enabled. Values are YES or NO.

Syntax

```
OSE.RESUME=YES|NO
```

A.1.4 BGSYNC.DISABLE

The `BGSYNC.DISABLE` parameter specifies whether the Sync Agent is disabled. Values are YES or NO. Disabling the Sync Agent prevents any automatic synchronization to be initiated for any user on this SQLite Mobile client.

Syntax

```
BGSYNC.DISABLE=YES|NO
```

A.2 POLITE.INI File Overview

The `POLITE.INI` file centralizes database volume ID assignments, defines parameters for all databases on a system, and defines synchronization parameters. On the WinCE platform, the file name is `POLITE.TXT`.

When you install SQLite Mobile client, the installation for Win32, WinCE and Linux platforms creates the `POLITE.INI` file in the *`<mobile_client_install_root>`*`\bin` directory.

Since the SQLite Mobile client configuration does not include the SQLite database, all sections except for the All Databases section applies on the SQLite Mobile client. Refer to Appendix E, "POLITE.INI Parameters" in the *Oracle Database Lite Administration and Deployment Guide* for details on these sections.

Index

A

Android
  application, 4-1, 4-7
    signed, 4-1, 4-7
  build, 4-3
  install, 4-1, 4-7
  prerequisites, 4-1
  sample, 4-2
  SDK, 4-1
  sign, 4-2
application
  Android, 4-1, 4-7
  install, 2-4
  overview, 1-2
  synchronization, 3-1
architecture
  client, 1-1
automatic synchronization
  disable, A-3

B

BGSYNC.DISABLE parameter, 3-7, A-3
BLOB
  queue
    naming, 3-6

C

CAB file
  registering, 2-9
  SDK version, 2-6, 2-9
client
  Android, 4-1, 4-7
  architecture, 1-1
  automatic synchronization, 2-12
  components, 1-1
  configuration, A-1
  custom platform, 2-9
  device management, 3-7
  file location, A-1
  install, 2-1, 2-5, 2-6
  language, 2-5, 2-7
  Linux, 2-4
  management, 3-1
  overview, 1-1
  platform, 1-1, 2-5, 2-7
  snapshots, 3-6
  software update request, 3-7
  start, 3-1
  synchronization, 1-1, 3-1
    GUI, 3-2
  Win32, 2-4
  Windows CE, 2-4
clock, 3-2
components
  client, 1-1
configuration
  parameters, A-1

D

database
  location, 2-12, A-1, A-2
  overview, 1-1
  support, 1-1
device
  client management, 3-7
  clock, 3-2
  install, 2-4
  multiple users, 2-8
device manager
  client, 3-7
  platforms, 1-2
  support, 2-1
dmagent, 1-2, 3-5

E

Eclipse, 4-1
  import project, 4-2

F

file-based synchronization
  enabling, 3-5
firewall, 3-8
force refresh, 3-4

I

In Queue
    naming, 3-6
INF file, 2-11
INI file
    description, 2-11
install
    Android, 2-6
    client, 2-1

M

MDK
    client compatibility, 2-6
Mobile client, see client
MOBILE_SYNC_ANDROID_LIB variable, 4-3
msync, 3-2, 3-5
    options, 3-4

N

network
    options, 3-3

O

operating system
    certified, 2-2
OSE.INI file, 3-6
    location, A-1
    overview, A-1
    parameters, A-1
OSE.RESUME parameter, 3-7, A-3
osync_android.jar file, 4-3
Out Queue
    naming, 3-6

P

password, 3-4
    reset, 3-5
performance
    resume feature, 3-7
platform
    client, 2-5, 2-7
    custom, 2-9
    requirements, 2-1, 2-2
    supported, 1-1, 2-1
polite.ini
    overview, A-3
    parameters, A-1
polite.txt
    description, A-3
priority
    synchronization, 3-4
private key
    Android, 4-7
proxy, 3-8
    reverse, 3-8
    specify, 3-3

R

resume feature
    configure, A-3
    manage, 3-7
reverse proxy, 3-8

S

sequence
    registry, 3-6
snapshots
    registry, 3-6
    types, 3-6, A-2
software
    request update, 3-7
SQLite database, see database
SQLite Mobile client, see client
SQLITE.DATA_DIRECTORY parameter, 2-12, A-2
SQLITE.QUEUES parameter, 3-6, A-2
Sync Agent
    disable, A-3
Sync Engine, 1-1
synchronization
    Android, 4-1, 4-2
        classes, 4-3
    APIs, 3-2
    automatic, 2-12
        disable, A-3
        manage, 3-7
        support, 2-1
    client, 1-1, 3-1
        GUI, 3-2
    customize, A-1
    file-based
        enabling, 3-5
    force refresh, 3-4
    msync, 3-2
    options, 3-4
    priority, 3-4
    Sync Engine, 1-1

T

transaction
    registry, 3-6

U

update utility, 3-5
user
    multiple users, 2-8
    password
        reset, 3-5