Oracle® Database Mobile Server

Mobile Client Guide

Release 11.1.0

E22681-02

September 2011

ORACLE®

Oracle Database Mobile Server Mobile Client Guide Release 11.1.0

E22681-02

Contents

Preface

This preface introduces you to the *Oracle Database Mobile Server 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

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documents

Use the following manuals and Web site as reference when installing and configuring Berkeley DB or SQLite:

- *Berkeley DB Installation and Build Guide*
- *Getting Started with the Oracle Berkeley DB SQL APIs*
- *http://www.sqlite.org/*

Conventions

The following conventions are also used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
boldface text	Boldface type in text indicates a term defined in the text, the glossary, or in both locations.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.
`italic monospace`	Italic monospace type indicates a variable in a code example that you must replace. For example: `Driver=`*`install_dir`*`/lib/libtten.sl` Replace *`install_dir`* with the path of your TimesTen installation directory.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

1 Mobile Client Overview

Oracle Database Mobile Server delivers critical bi-directional data synchronization capability to mobile or fixed location distribution devices, while providing a centralized backend interface for managing mobile deployments. On the client device, the mobile client facilitates the transfer of data to and from the client database, which can be either Berkeley DB or SQLite. Install the desired database and the mobile client for Berkeley DB or SQLite on your client device.

The following sections describe both databases and the mobile client for these databases:

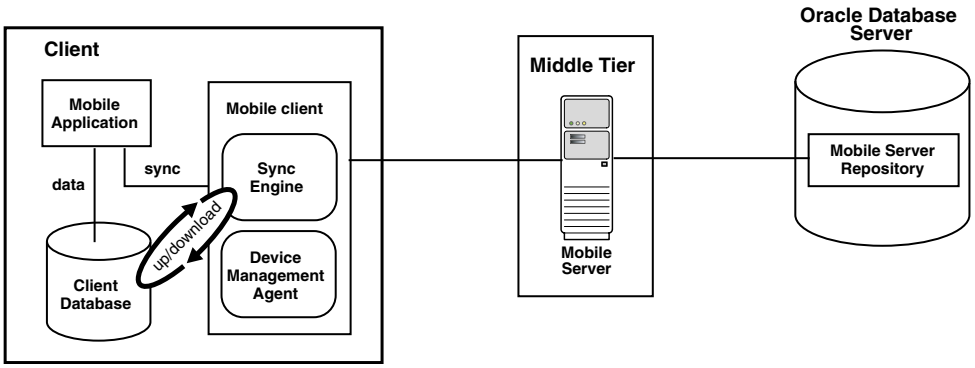
1.1 Mobile Client Architecture

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

- Client database—The client database can be either Berkeley DB or SQLite, which is installed independently of the mobile client.
- Mobile client—When you install the mobile client, the following components are provided:
  - Sync Engine—Automatic synchronization can be enabled on the Blackberry, Android, Win32, WinCE, and Linux platforms. However, you can initiate manual synchronization within a mobile application on all platforms.

    The Sync Engine interacts with SQLite 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 mobile server uses the DM Agent to send commands to the mobile device for remote management. The DM Agent is only installed on Android, Win32, WinCE, and Linux platforms. The Blackberry platform cannot be remotely managed.
- Mobile application—Interacts with the client database to manage the data. Interacts with the Sync Engine to initiate a manual synchronization.

Figure 1–1 Architecture for Device with a Mobile Client and Client Database



The following sections describe each mobile client:

- Section 1.2, "Mobile Client for the Berkeley DB SQL API Interface"
- Section 1.3, "Mobile Client for SQLite"

1.2 Mobile Client for the Berkeley DB SQL API Interface

Berkeley DB is a general-purpose, high-performance, embedded database that is designed for high-throughput applications. The primary goal of Berkeley DB is to deliver fast, scalable and flexible data management services to your application while remaining transparent to the end-user. Berkeley DB executes in the same process as your application.

Berkeley DB provides the following features that are expected of client/server enterprise-scale SQL databases: high throughput, high availability, high concurrency, replication, low-latency reads, non-blocking writes, failure recovery, data scalability, in-memory caching, ACID transactions, automatic and catastrophic recovery. Berkeley DB offers advanced features in a self-contained, small footprint software library.

The mobile client was built to use the Berkeley DB SQL interface, which combines the programming interface of SQLite with the Berkeley DB storage engine. The mobile client uses this interface to facilitate synchronization between the client and the back-end database.

The following sections describe the Berkeley DB SQL interface and how it is used to synchronize data with the mobile client:

- Introducing the Berkeley DB SQL Interface
- Synchronizing Data from Berkeley DB to a Back-End Oracle Database

1.2.1 Introducing the Berkeley DB SQL Interface

The Berkeley DB SQL interface comes with a SQL processor layer on top of Berkeley DB. The Berkeley DB SQL interface is API compatible with SQLite, so it can be used as a replacement for SQLite applications. Thus, you can manage relational data in Berkeley DB, but access the data with the SQLite3 API.

The interaction with the Berkeley DB SQL interface is almost identical to SQLite. You can use the same APIs, SQL statements, command shell environment, and most of the PRAGMAs with the Berkeley DB SQL interface. There are no differences in the SQL data types between the Berkeley DB SQL API and the SQLite API.

The documentation for the Berkeley DB SQL interface is in two books within the Berkeley DB documentation:

- *Berkeley DB Installation and Build Guide*—The SQL interface is not installed by default. You must explicitly request it to be included when building Berkeley DB. The "Building the SQL API" section in this book details the correct build steps for the Berkeley DB SQL interface. It also includes information on compatibility with SQLite.

  This section can be accessed at the following site:

  `http://download.oracle.com/docs/cd/E17076_02/html/installation/build_win_sql.html`

- *Getting Started with the Oracle Berkeley DB SQL APIs*—The main book for the Berkeley DB SQL interface, which can be accessed at the following site:

  `http://download.oracle.com/docs/cd/E17076_02/html/bdb-sql/index.html`

The Berkeley DB product and all documentation is available at the following site:

`http://www.oracle.com/technetwork/database/berkeleydb/overview/index.html`

The default installed BDB SQL interface DLLs and the command line interpreter are named as follows:

- `dbsql.exe` on WIndows and `dbsql` on UNIX—This is the command line shell. It operates identically to the SQLite shell, `sqlite3.exe` on Windows and `sqlite3` on UNIX.
- `libdb_sql50.dll` on Windows and `libdb_sql` on UNIX—This is the library that provides the BDB SQL interface. It is the equivalent and compatible with the SQLite library, `sqlite3.dll` on Windows and `libsqlite3` on UNIX.

If you want the names to be exactly the same names as SQLite, you can perform the following:

- On Windows, copy `dbsql.exe` to `sqlite3.exe` and `libdb_sql51.dll` to `sqlite3.dll`. Once copied, you can use these applications as a replacement for the standard SQLite binaries with the same names.
- On UNIX, specify the compatibility option (`--enable-sql_compat`) for the BDB SQL interface UNIX build.

However, this must only be performed with extreme caution. For more details on building the BDB SQL Interface to have the same names as SQLite, see "Building the SQL API" section in the "Berkeley DB Installation and Build Guide" in the Berkeley DB documentation.

1.2.2 Synchronizing Data from Berkeley DB to a Back-End Oracle Database

The Berkeley DB Mobile Client interacts with Berkeley DB, which must be installed with its Berkeley DB SQL interface. The mobile client synchronizes the data in Berkeley DB with the mobile server. This book describes how to configure, manage and implement synchronization using the mobile client. It does not discuss how to build, install, configure, manage or use the Berkeley DB SQL interface.

1.3 Mobile Client for SQLite

SQLite 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 SQLite already installed, including Android and Blackberry devices.

You can synchronize the data in one or more SQLite databases to a back-end Oracle database with the mobile client. This mobile client provides the ability to synchronize the data in SQLite databases with the Sync Engine contained within the mobile client.

SQLite is installed independently from the mobile client. SQLite does not provide the same SQL functionality as an Oracle database. This book describes how to configure, manage and implement synchronization using the mobile client. It does not discuss how to configure, manage or use SQLite. For information on SQLite and a full list of what functionality is supported, see http://www.sqlite.org/.

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

2

Installing the Mobile Client

One of the benefits of Oracle Database Mobile Server 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 mobile client, Oracle Database Mobile Server installs the Sync Engine and Device Manager.

> **Note:** Every mention of the mobile client in this book refers to both the Berkeley DB Mobile Client and the SQLite Mobile Client.
>
> Each mention of Berkeley DB refers to the Berkeley DB SQL interface.

The following sections detail how to install the mobile client software on your client device:

- Section 2.1, "Supported Platforms and Requirements for the Mobile Client"
- Section 2.2, "Preparing the Device for a Mobile Application"
- Section 2.3, "Installing the Mobile Client"
- Section 2.4, "Configuring the Location of Mobile Client and Database Files"
- Section 2.5, "Configuring for Automatic Synchronization When Installing the Client"
- Section 2.6, "Uninstalling the Mobile Client"

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

2.1 Supported Platforms and Requirements for the Mobile Client

The Berkeley DB and SQLite Mobile Clients are certified for SUN JDK 1.6 for the Java APIs and can be installed on the following platforms:

- Android 1.6, 2.1, 2.2, 2.3, and 3.0
- Blackberry RIM 5.0 and 6.0
- Microsoft Windows 2003 (32-bit and 64-bit)
- Microsoft Windows 2008 (32-bit and 64-bit R2)
- Microsoft Windows XP Professional Edition with Service Pack 3, 32-bit
- Microsoft Windows 7 (32-bit and 64-bit)
- Microsoft Windows Vista Ultimate

- Oracle Enterprise Linux 5.0, or 6.0 (32-bit and 64-bit)

You can also install Berkeley DB and SQLite Mobile Clients on the Microsoft Windows Mobile 6 and 6.5 platforms.

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

Automatic synchronization and device management are available on most mobile client platforms. Table 2–1 displays what features are available on which platforms.

Table 2–1 Feature Support for Client Platforms

Platform	Automatic synchronization	Device management through the DM Agent
Blackberry	Yes	No
Android	Yes	Yes
Win32	Yes	Yes
WinCE	Yes	Yes
Windows Mobile	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 mobile client. The requirements do not include requirements for either client database, but are only the requirements for the mobile client including the Sync Engine and Device Manager.

- 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 Storage for Mobile Client
BlackBerry	100 KB
Android	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 2003, Windows 2008, Windows Vista Ultimate, Windows XP Professional Edition with Service Pack 2, Windows 7	If using Java APIs for synchronization, use Sun JDK 1.6. If implementing any .NET applications, use Compact Framework .NET 1.1 or 2.0

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

Device Platform	Certified Operating System	Other Software Requirements
Windows CE	Windows CE 5.0 See Table 2–5, " Pocket PC and Windows Mobile Supported Platforms" for full details.	If using Java APIs for synchronization, use Sun JDK 1.6. ActiveSync version 3.8 or higher. Microsoft .NET Compact Framework 1.0
Windows Mobile	▪ Windows Mobile 5 ▪ Windows Mobile 5 for Pocket PC ▪ Windows Mobile 5 for Pocket PC Phone Edition ▪ Windows Mobile 5 AKU2	If using Java APIs for synchronization, use Sun JDK 1.6.. 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 Java APIs for synchronization, use Sun JDK 1.6.. 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 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`.

For both the Berkeley DB and SQLite Mobile Clients, ADO.Net is supported by SQLite. For more information on ADO.Net support, see the following URL:

`http://system.data.sqlite.org/index.html/doc/trunk/www/features.wiki`

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

Device Platform	Supported Technologies	Certified Technologies
Win32	▪ ADO.Net ▪ JDBC ▪ ODBC	Sun Microsystems Java Runtime Edition 5.0 and 6.0
Windows CE	▪ Microsoft ActiveSync version 3.8 or for Windows CE 5.0, use Microsoft ActiveSync version 4.1 or higher. ▪ ADO.Net ▪ JDBC ▪ ODBC	Sun JDK 1.6.
Linux x86	JDBC	Sun Microsystems Java Runtime Edition 5.0 and 6.0

For each native platform, a CAB file is downloaded from the setup page. The naming structure for each CAB file is as follows: `<mobile_client>.<language>.<platform>.<cpu>.cab`, where `<mobile_client>` can be `bdb` or `sqlite`.

Table 2–5 Pocket PC and Windows Mobile Supported Platforms

Product Name	WinCE Version	Chipsets	SQLite Client CAB file download from Setup page
Windows Mobile 5 and Windows Mobile 5 AKU2	5.0 and 5.1.465	ARMV4I	PPC50 ARMV4I, which uses the `<mobile_client>.<language>.ppc50.armv4i.cab`
Windows Mobile 6	5.2.1236	ARMV4I	PPC60 ARMV4I, which uses the `<mobile_client>.<language>.ppc60.armv4i.cab`

2.2 Preparing the Device for a Mobile Application

To execute mobile applications on a device, do the following:

1. Install the 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 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 user name/password of the mobile user who owns the mobile applications. The data for each application is retrieved.

Notes: For the restrictions on creating the user name and password, see Section 4.3.1.2.1, "Define User Name and Password" in the *Oracle Database Mobile Server Administration and Deployment Guide*.

For more information about synchronization, see Chapter 5, "Managing Synchronization" in the *Oracle Database Mobile Server Administration and Deployment Guide*.

---

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

2.3 Installing the 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.
- While you can synchronize multiple Berkeley DB or SQLite databases on the same client, you cannot synchronize both SQLite and Berkeley DB on the same mobile client.

The following sections provide directions for the mobile client install:

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

2.3.1 Installing the Mobile Client on Blackberry Devices

To install the 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>/mobile/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 mobile client for your language and the Blackberry client platform. This downloads and installs the mobile client.
3. Perform a manual synchronization for the mobile client.
4. Synchronization requires you to enter the user name 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 user name and password, see Section 4.3.1.2.1, "Define User Name and Password" in the *Oracle Database Mobile Server Administration and Deployment Guide*.

2.3.2 Installing the 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 mobile client binaries unless they are downloaded within the context of a signed application.

Thus, the instructions for installing the mobile client on Android devices is presented with an example of creating and downloading the Android application on a SQlite Mobile Client, which is provided in Chapter 4, "Using an Android Application on the SQLite Mobile Client".

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

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

To install the 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, Windows Mobile or Linux client, see Section 7.1, "Customize the Mobile Client Software Installation for Your Mobile Device" in the *Oracle Database Mobile Server Administration and Deployment Guide*.

1. On the mobile client, open a browser to point to the mobile server using the following URL.

   `http://<mobile_server>:<port>/mobile/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.

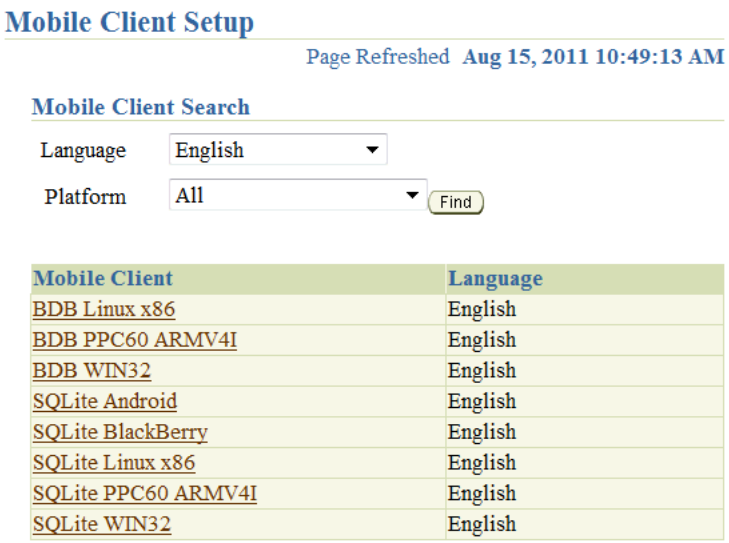
   - Language: Select a language other than English on the Language pulldown. English is the default.
   - Platform: Choose to see all available platforms for the indicated language.

   Client platforms are provided in the mobile client setup page. These client CAB files are optimized for size to minimize the footprint on your device. For each native platform, a CAB file is downloaded from the setup page. The naming structure for each CAB file is *<mobile_client>.<language>.<platform>.<cpu>*`.cab`, where *<mobile_client>* can be `bdb` or `sqlite`.

   However, if you are using a client with the Standard SDK for WinCE 5.0 platforms for Windows Mobile 5 (WCESTDSDK), 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 Windows Mobile Files for Your Mobile Client".

Figure 2–1 Mobile Client Setup Page



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

2. Click the 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 Mobile Server Windows Mobile platforms to ActiveSync. Then, when the device is put into the cradle, ActiveSync installs the Oracle Database Mobile Server 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 user name and password for the mobile user.

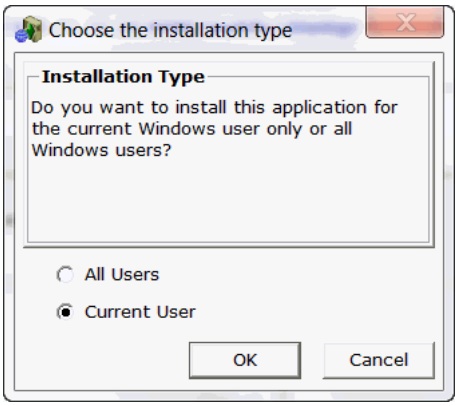
Note: For the restrictions on creating the user name and password, see Section 4.3.1.2.1, "Define User Name and Password" in the *Oracle Database Mobile Server Administration and Deployment Guide*.

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 Mobile Server Administration and Deployment Guide*.

Figure 2–2 Select Installation Privileges



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

Mobile Client	Initial Synchronization Details
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 Mobile Server on the device. 2. After the mobile client is installed on the device, start the Device Manager Agent on the device either by selecting Device Manager in the programs group or by executing `dmagent.exe`, which is in the `orace` directory. 3. Enter the user name and password. If the mobile server URL field is empty, provide the URL as well. 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 Windows Mobile Files for Your Mobile Client

By default, the Windows Mobile 5 (PPC50) and Windows Mobile 6, 6.1 and 6.5 (PPC60) 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 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 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 CAB files are named *<mobile_client>.<language>.<platform>.<cpu>*.cab. Rename the CAB file to bdb.cab or sqlite.cab as appropriate and copy it into a subdirectory according to language and client platform type relative to the *<app_server_deployment_dir>*\mobile\setup directory. Take note of the directory path as you will provide the location of the CAB 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.

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 *<app_server_deployment_dir>*\Mobile\Setup\devmgr 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 sqlite_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 7.9, "Installation Configuration (INF) File"in the *Oracle Database Mobile Server 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.cab</src>
    <des>$APP_DIR$\sqlite.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:

<*app_server_deployment_dir*>\Mobile\Setup\devmgr\wince\<platform>\<chipset>\

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

<*app_server_deployment_dir*>\Mobile\Setup\devmgr\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_SQLite_60.
- INF: Provide the name of the INF file, such as sqlite_win32.inf or sqlite_linux-x86.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 Mobile Server 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_SQLite_60
INF=sqlite_std500.inf
BOOTSTRAP=DeviceInfo
ATTRIBUTES=dmc=auto&update=true&enabled=true
```

2.4 Configuring the Location of Mobile Client and Database Files

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

- All client databases and temporary synchronization data are stored in the `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 mobile client is stored in the `DATA_DIRECTORY/oseconf` directory.

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

```
[SQLITE]
DATA_DIRECTORY=C:\mobileclient\sqlite
```

For more details on this parameter, see Appendix A.1.2.1, "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 7.5, "Sending Commands to Your Mobile Devices" in the *Oracle Database Mobile Server Administration and Deployment Guide* for more details on sending commands to your mobile device.

2.6 Uninstalling the Mobile Client

When you want to uninstall the mobile client, execute the `uninst.exe` that is located in the install directory for the mobile client.

3

Managing Your Mobile Client

The following sections describe how to manage the Oracle Database Mobile Server functionality on the mobile client:

3.1 Starting the Mobile Client

When you installed the mobile client on Linux or Windows, it is configured so that the mobile client always starts automatically when the device is initiated.

3.2 Synchronize Data for Applications on the Mobile Client

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

The following describes how to initiate synchronization from each type of 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 4, "Managing Synchronization on the Mobile Client" in the *Oracle Database Mobile Server 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:

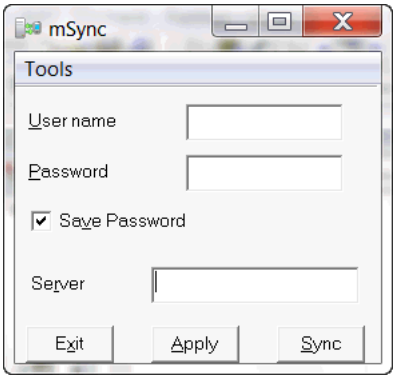
  - Execute the `msync` executable, described in Section 3.3, "Use the mSync GUI to Initiate Synchronization".
  - Implement synchronization within your application using the synchronization APIs, as described in Chapter 2, "Synchronization" and Chapter 4, "Managing Synchronization on the Mobile Client" in the *Oracle Database Mobile Server 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 user name or password. Please enter correct value and reSync."`

3.3 Use the mSync GUI to Initiate Synchronization

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

Figure 3–1 Using the mSync GUI to Initiate Synchronization



To bring up the mSync GUI, execute `msync.exe` on WinCE and Win32 or `msync` on Linux, which is located in the `/mobileclient/bin` 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:

- User name and password for the user that is starting the synchronization.

> **Note:** See Section 4.3.1.2.1, "Define User Name and Password" in the *Oracle Database Mobile Server Administration and Deployment Guide* for conventions for creating the user name or password.

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

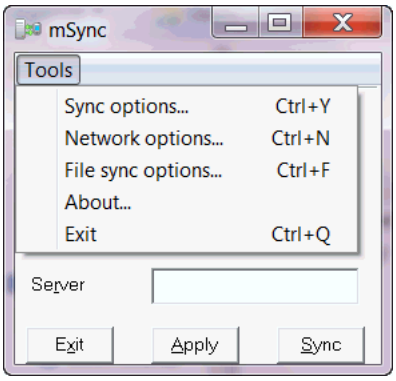
Click **Sync** to start the Synchronization. Click **Apply** to save any modifications you made to the entries. Click **Exit** to leave the tool.

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 3.10, "Initiate Updates for the 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 menu, as shown in Figure 3–2.

Figure 3–2 The mSync Tools Selection



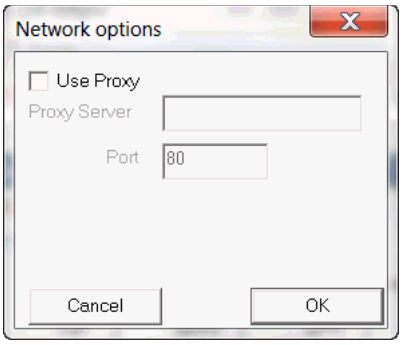
The following sections describe the Tools options:

- 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 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

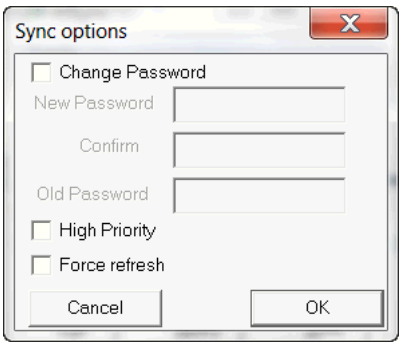
Figure 3–4 displays the Sync Options screen where you can specify the following:

- 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 Mobile Server 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



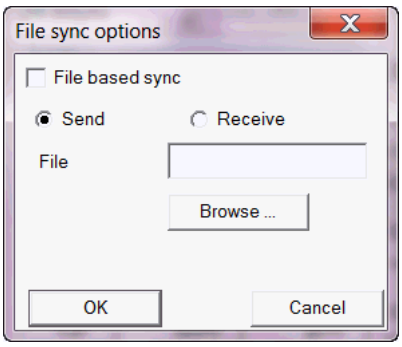
3.3.3 Sync to a File Using File-Based Sync

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 5.10, "Synchronizing to a File with File-Based Sync" in the *Oracle Database Mobile Server Administration and Deployment Guide*.

Figure 3–5 File Sync Options



3.3.4 Use Mobile Client Tools on Linux

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

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 server 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 9.2, "Which Password is Which" in the *Oracle Database Mobile Server Administration and Deployment Guide* for details on passwords.

3.5 Manage Snapshots on the Mobile Client

The following are the types of snapshots you can enable for tracking the changes on the client 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 mobile client Java APIs only support state-based snapshots. To enable state-based snapshots, set the `QUEUES` parameter in the `OSE.INI` file to `NO`.

  Snapshot state tables, `OSE_ST$<snapshot>`, are created in the client 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 client 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 `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.1.2.2, "QUEUES".

3.6 Control Automatic Synchronization for a Specific Mobile Client

As described in Section 5.5, "Using Automatic Synchronization" in the *Oracle Database Mobile Server 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 DISABLE parameter to YES in the OSE.INI file on the 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.1.3, "Background Sync Parameter—BGSYNC".

3.7 Providing Security for the Mobile Client

The introduction of handheld devices within the corporate environment can pose a security threat to an organization. Devices are now used to store not only company contacts; but, with external cards, may store up to 60 gigabytes of information or more. Devices also provide a mobile point of entry into the organizational network that is located outside the network security perimeter. It is essential to secure this data if a device is lost or compromised.

Securing a device involves a layered approach. You must secure not only access to the device, but data stored on the device and communications across the network. Most aspects of security for a mobile device must be incorporated before Oracle Database Mobile Server is included within the security infrastructure.

1. Security starts with the device itself. Authentication on the device must be implemented through pin or password authentication, biometric readers, secure digital media for storage, and even how the device is stored, transported, and accounted for.
2. Once access is gained to the device, further security must be implemented within the mobile application to prevent the application from being able to retrieve invalid data. Technologies, such as the Microsoft.Net Compact Framework, incorporate API calls that may be used to encrypt and decrypt any data that will be stored or retrieved from the device.

Oracle Database Mobile Server provides several security features that may be utilized to help in securing data. These features aid in protecting information during synchronization and once access to a device has been obtained. The two most important aspects of security for the mobile infrastructure are the following:

1. Use Secure Socket Layer (SSL) to protect the transmission of data during the synchronization process. For full details, see Section 9.4, "Configuring for Secure Socket Layer (SSL) Communication" in the *Oracle Database Mobile Server Administration and Deployment Guide*.

2. Encrypt the mobile client database, whether for the Berkeley DB or the SQLite database. For more details, see Section 3.7.1, "Encryption for the Berkeley DB and SQLite Databases".

3.7.1 Encryption for the Berkeley DB and SQLite Databases

For both the Berkeley DB and SQLite databases, you can encrypt the data by using the encryption methods provided by Berkeley DB and SQLite respectively. For details on encryption for these databases, see the following:

- SQLite provides a proprietary extension for encryption called SQLite Encryption Extension (SEE). For more information, see the following link:

  `http://www.hwaci.com/sw/sqlite/see.html`

- The Berkeley DB SQL Interface also supports the SQLite Encryption Extension (SEE) with some limitations. Berkeley DB encryption is discussed in the following documentation:

  `http://download.oracle.com/docs/cd/E17076_02/html/bdb-sql/sql_encryption.html`

3.8 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 mobile client by setting the `RESUME` parameter in the `OSE.INI` file to NO. For more details on the resume feature and disabling it for your mobile client, see Section A.1.1, "Resume Parameter—OSE" and Section 5.7, "Resuming an Interrupted Synchronization" in the *Oracle Database Mobile Server Administration and Deployment Guide*.

3.9 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 7.7, "Using the Device Manager Agent (dmagent) on the Client" in the *Oracle Database Mobile Server Administration and Deployment Guide* for a full description.

3.10 Initiate Updates for the Mobile Client

You can initiate a request for software updates from the mobile server by executing the Oracle Database Mobile Server Update tool. For details, see Section 7.6.3, "Initiate Updates of Oracle Database Mobile Server Software for Mobile Clients" in the *Oracle Database Mobile Server Administration and Deployment Guide*.

3.11 Communicate Between the Internet and Intranet Through a Reverse Proxy

If a Win32, WinCE or Linux mobile client is on either side of the firewall, set up a proxy or reverse proxy to facilitate communication between the mobile client and mobile server. See Section 9.6, "Using a Firewall Proxy or Reverse Proxy" in the *Oracle Database Mobile Server Administration and Deployment Guide*.

4

Using an Android Application on the SQLite Mobile Client

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

> **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.

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 Mobile Development Kit.

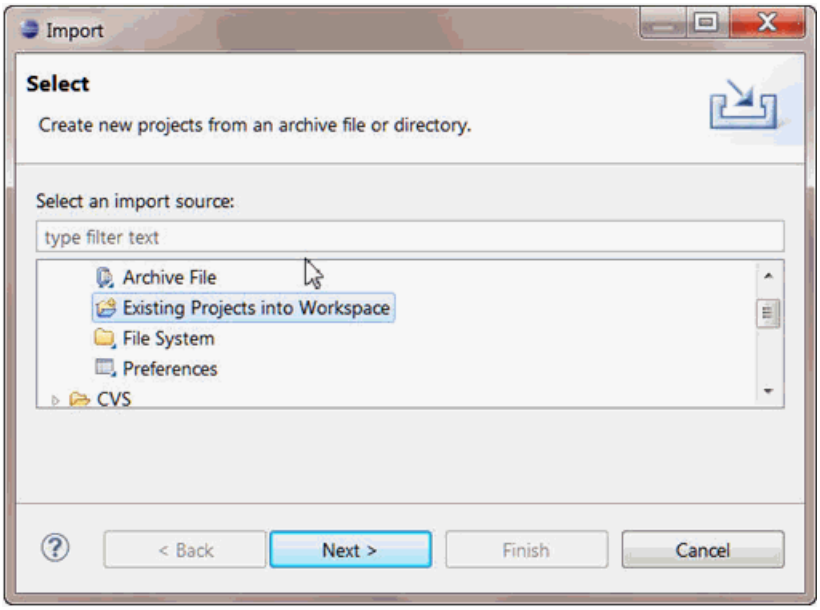
4.2 Import the Oracle Database Mobile Server Android Project into Eclipse

Import the Oracle Database Mobile Server `simple_sync_android` sample Android project into your Eclipse Workspace.

The following steps show how to import the mobile server sample Android project.

1. In Eclipse, with your Workspace open, select File->Import and choose **Existing Projects into Workspace**. Click **Next**.

Figure 4–1 Import Existing Projects into Eclipse Workspace



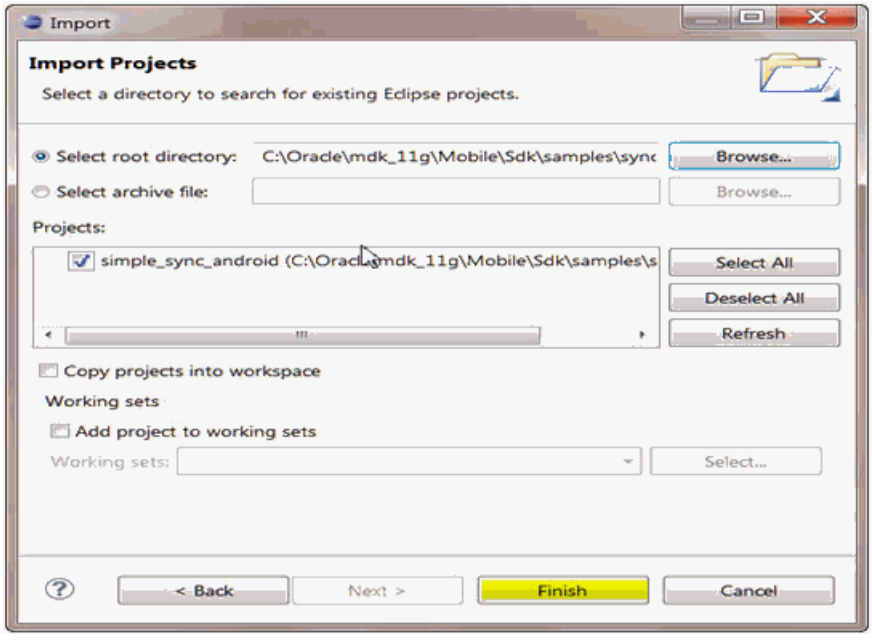
2. Set the root directory to point to the Android project within the Oracle Database Mobile Server MDK. Enable the **Select Root Directory** button and browse for the `simple_sync_android` project, which is located in the following directory:

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

   where the *<MDK_ROOT>* is replaced with the full path where the Oracle Database Mobile Server 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



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

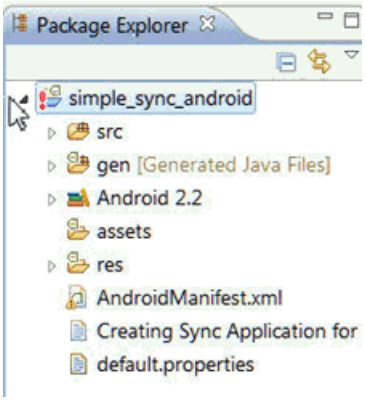
4.3 Build Oracle Database Mobile Server Android Project

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

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

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

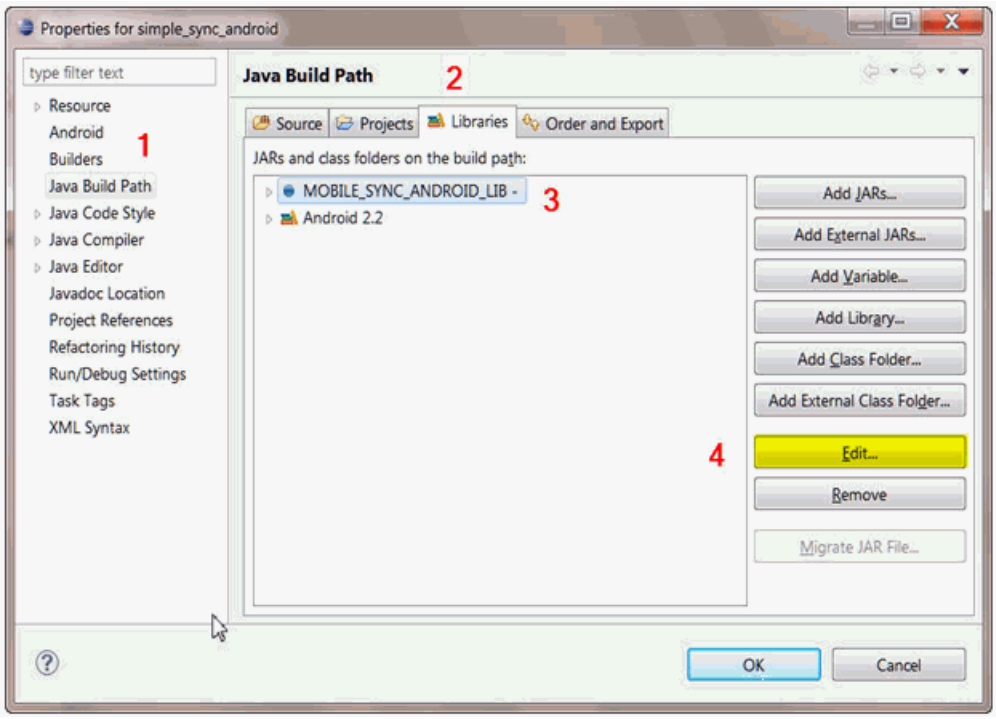
Figure 4–3 Project Explorer window



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

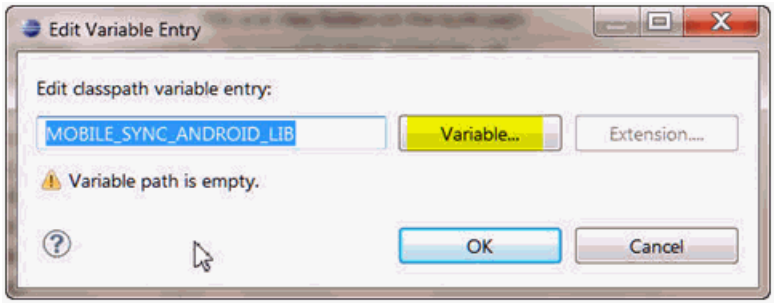
   c. As shown in Figure 4–4, select the **Java Build Path** in the left pane. Then, select the **Libraries** tab in the Build Path window. Select **`MOBILE_SYNC_ANDROID_LIB`** to configure the location of the JAR file. The `MOBILE_SYNC_ANDROID_LIB` variable provides the library directory where the `osync_android.jar` file is located, which is required to build your project. Click **Edit**.

Figure 4–4 Properties for simple_sync_android Project



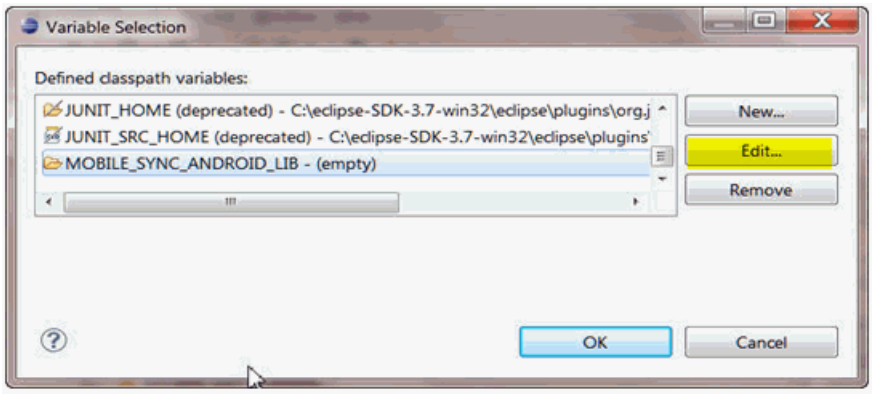
d. In the "Edit Variable Entry" dialog, click the **Variable** button.

Figure 4–5 Edit the MOBILE_SYNC_ANDROID_LIB Variable



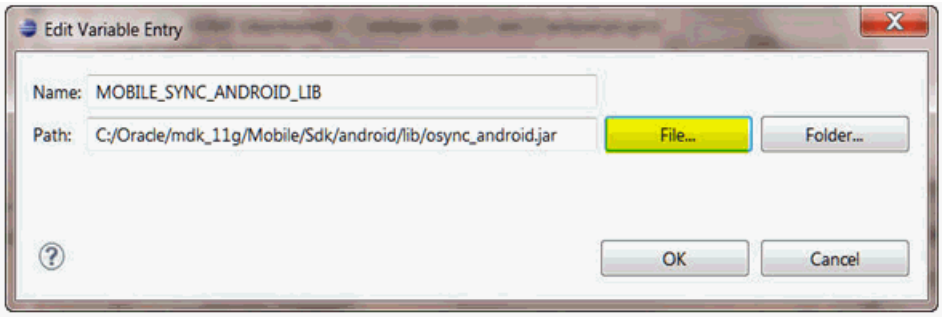
e. Click **Edit**.

Figure 4–6 Edit Path Variable



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

Figure 4–7 Add Path to the MOBILE_SYNC_ANDROID_LIB Environment Variable



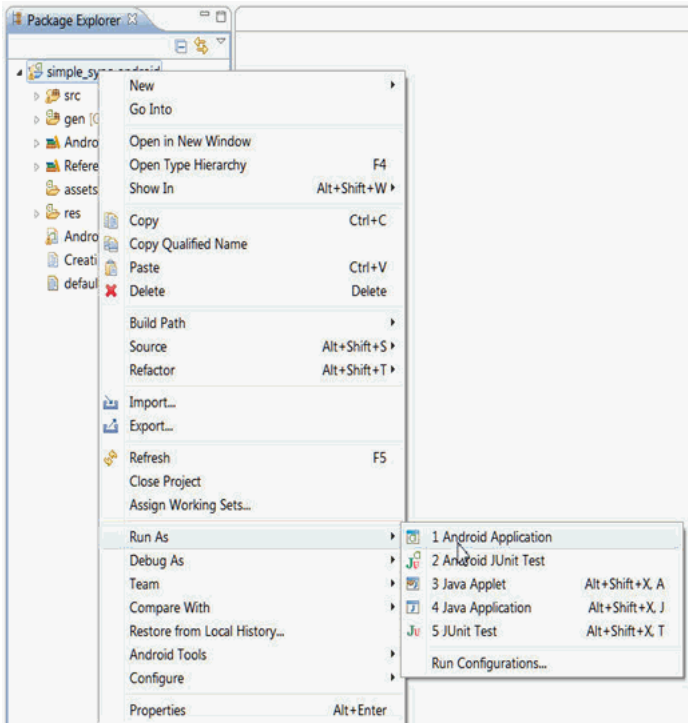
g. When finished, click **OK**.

h. In the "Classpath Variables Changed" dialog, select **Yes** to rebuild the project.

Figure 4–8 Classpath Variables Changed

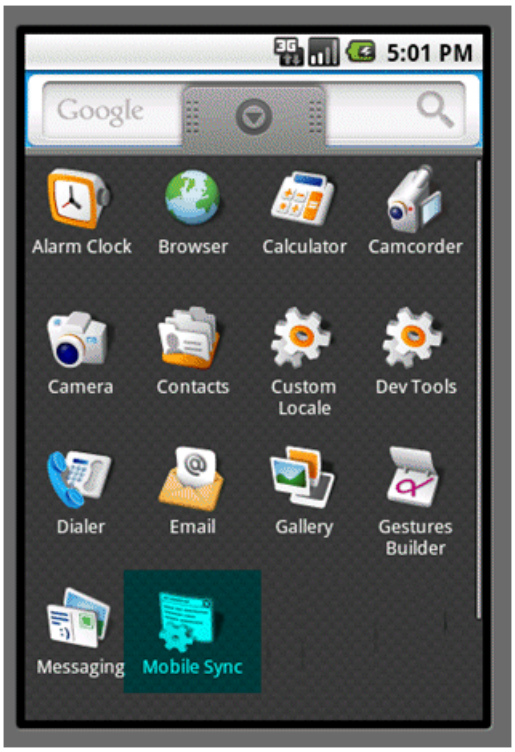
2. Build the `simple_sync_android` project to link in the `osync_android.jar` file with the Oracle Database Mobile Server synchronization libraries.
3. Execute and debug the `simple_sync_android` project with the Android emulator.
   a. Right click on the `simple_sync_android` project.
   b. Click **Run As** and select **Android Application**.

Figure 4–9 Executing the Android Application



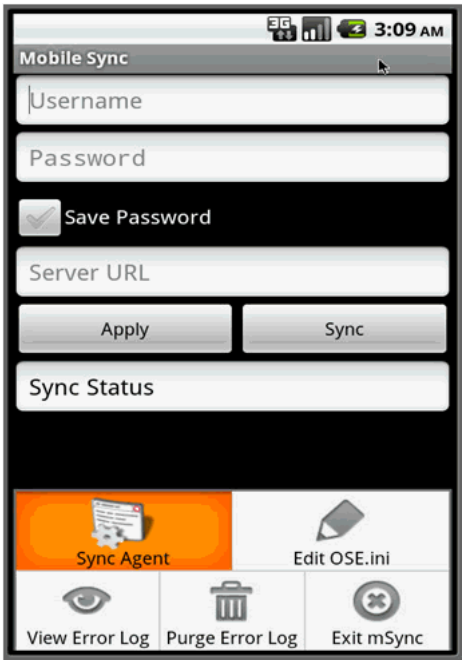
The Android emulator is started, where you can execute the sample as the **Mobile Sync** application.

Figure 4–10 Test Project with Android Emulator



4. When you execute the Mobile Sync application, synchronization is initiated. Enter your user name, password and mobile server URL in the Sync UI, as shown in Figure 4–11. Click **Sync** to start the synchronization.

Figure 4–11 Synchronization UI



5. Examine the `MainAct.java` code. The sample consists of the UI layout code that is located in the `res\layout` subdirectory. The main application Java class file is `MainAct.java`, which contains all of the logic for initializing synchronization structures and invoking the synchronization methods. You can examine this file for more details on the sequence of calls for the synchronization APIs.

A

Mobile Client Configuration Parameters

You can customize the mobile client by modifying the parameter values defined in the OSE.INI configuration file.

The installation automatically sets the parameters in the OSE.INI file, 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 sections detail the parameters within the OSE.INI and DEVMGR.INI configuration files:

- Section A.1, "OSE.INI File Overview"
- Section A.2, "DEVMGR.INI File"
- Section A.3, "Sample OSE.INI and DEVMGR.INI Files"

A.1 OSE.INI File Overview

The OSE.INI file stores properties used by the mobile clients. It contains parameters that define the location of the mobile client database and mobile client files, defines parameters for all databases on a system, and how to customize synchronization for the mobile client 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. On WinCE and EPOC, the file name is OSE.TXT.

> **Note:** The installation automatically sets the parameters in your OSE.INI file, but you can customize the mobile client by modifying the parameter values defined in your OSE.INI file, which is available in Windows under %WINDIR%\OSE.INI and in Linux under $ORACLE_HOME/bin. You must have write permissions on the directory where this file is located to be able to modify the OSE.INI file. To modify the OSE.INI file, use an ASCII text editor.

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

- On Win32, WinCE and Linux platforms, the `OSE.INI` file is located in the *<mobile_client_install_root>*`\bin` directory.
- On SQLite 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 parameter sections for the `OSE.INI` file:

- Section A.1.1, "Resume Parameter—OSE"
- Section A.1.2, "SQLite Mobile Client Parameters—SQLITE"
- Section A.1.3, "Background Sync Parameter—BGSYNC"

A.1.1 Resume Parameter—OSE

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

Syntax

```
[OSE]
RESUME=YES|NO
```

A.1.2 SQLite Mobile Client Parameters—SQLITE

The `SQLITE` section configures certain aspects of both the Berkeley DB and SQLite Mobile Clients. The following sections describe the mobile client parameters that you can modify:

- Section A.1.2.1, "DATA_DIRECTORY"
- Section A.1.2.2, "QUEUES"
- Section A.1.2.3, "LIMIT_CONNECTIONS"

A.1.2.1 DATA_DIRECTORY

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

- SQLite, Oracle Database Mobile Server repository files, and temporary synchronization data are stored in the `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 `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.1.2.2 QUEUES

The QUEUES parameter specifies which type of snapshots the client will use in tracking the changes for Berkely DB and SQLite databases. 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 Berkely DB and SQLite databases 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 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 QUEUES parameter in the OSE.INI file to NO.

Syntax

```
[SQLITE]
QUEUES=YES|NO
```

A.1.2.3 LIMIT_CONNECTIONS

Set LIMIT_CONNECTIONS to YES when you want to limit the number of concurrent connections used by synchronization. Setting this parameter to YES keeps alive only the minimum required number of connections. If the QUEUES parameter is set to YES, the minimum number of connections necessary for synchronization is 2. If QUEUES is set to NO, only a single connection is required.

Setting the LIMIT_CONNECTIONS parameter is a trade-off between performance and memory limitations. This parameter is set to YES by default on all Blackberry devices for conserving memory.

Syntax

```
[SQLITE]
LIMIT_CONNECTIONS=YES|NO
```

A.1.3 Background Sync Parameter—BGSYNC

The 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 DEVMGR.INI File

The DEVMGR.INI file contains mobile client parameters for Device Management in the DMC section and the network parameters in the NETWORK section. For full details on device management parameters that can be modified before installing the client, see Section 7.2, "Configuring Mobile Clients Before Installation" in the *Oracle Database Mobile Server Administration and Deployment Guide*.

The following sections describe the parameters for the DMC and Network sections:

- Section A.2.1, "Device Management Parameters—DMC Section"
- Section A.2.2, "Network Parameters—NETWORK Section"

A.2.1 Device Management Parameters—DMC Section

The Device Management parameters are as follows:

- Section A.2.1.1, "DISABLE_PROMPT"
- Section A.2.1.2, "PUSH_PORT"
- Section A.2.1.3, "UPDATE_DAY and UPDATE_TIME"
- Section A.2.1.4, "MAX_RETRY"
- Section A.2.1.5, "FREQUENCY"
- Section A.2.1.6, "DEBUG"

A.2.1.1 DISABLE_PROMPT

The DISABLE_PROMPT parameter accepts a TRUE or FALSE value, which causes the following action:

- TRUE: The device checks for software updates available on the server. If updates are available, these are brought down to the client and installed.
- FALSE: The device checks for software updates available on the server. If updates are available, the option to bring down the updates and install them is displayed to the user, who decides what action to take. If the client chooses to update, then these are brought down to the client and installed.

A.2.1.2 PUSH_PORT

The port number on the mobile device that accepts device management commands from the mobile server. By default, the port number is 8521. Do not modify on the client. Even though it is described here, you should only modify the PUSH_PORT variable in the INF file BEFORE the mobile client is installed. For full details, see Section 7.2, "Configuring Mobile Clients Before Installation" in the *Oracle Database Mobile Server Administration and Deployment Guide*.

A.2.1.3 UPDATE_DAY and UPDATE_TIME

The day and time to check for software updates for the client. You can modify day and time here or within the DMAgent UI. For details on the DMAgent UI, see Section 7.7, "Using the Device Manager Agent (dmagent) on the Client" in the *Oracle Database Mobile Server Administration and Deployment Guide*. If you do want to modify them here, the values are as follows:

Day when the device checks for software updates. Used in combination with `UPDATE_TIME`.

UPDATE_DAY takes 0 - 8 which translates to the following days:

- Never = 0
- Daily = 1
- Sunday = 2
- Monday = 3
- Tuesday = 4
- Wednesday = 5
- Thursday = 6
- Friday = 7
- Saturday = 8

Time of day that the device checks for software updates from the mobile server. Used in combination with `UPDATE_DAY`. `UPDATE_TIME` can take values 0 - 23 which translates to the following time:

- 00:00 = 0
- 01:00 = 1
- 12:00 = 12
- 13:00 = 13
- 23:00 = 23

A.2.1.4 MAX_RETRY

Integer value that configures the maximum number of retry attempts before abandoning a server command.

A.2.1.5 FREQUENCY

The frequency of how many seconds between the client polls. The DMAGENT connects to the mobile server checking for new commands at the defined `FREQUENCY` interval.

A.2.1.6 DEBUG

If you turn on the `DEBUG` parameter in the `[DMC]` section, then this turns on the debugging for the device manager. All device manager debug messages are written to the `_dmdebug.txt` file.

To enable, set the `DEBUG` parameter in the `[DMC]` section to 1. Set to 0 to turn off debug feature, which is the default.

Default value: 0

A.2.2 Network Parameters—NETWORK Section

The following parameter configures how the client interacts over the network:

- Section A.2.2.1, "SERVER_URL"
- Section A.2.2.2, "DISABLE_SSL_CHECK"
- Section A.2.2.3, "HTTP_PROXY"

A.2.2.1 SERVER_URL

This parameter points to the mobile server. It communicates with the mobile server over HTTP or HTTPS. The expected syntax for the `SERVER_URL` parameter is as follows:

```
HTTP://<host>:<port>/mobile
```

For example:

```
[NETWORK]
SERVER_URL=HTTPs://myhost:8888/mobile
```

A.2.2.2 DISABLE_SSL_CHECK

You can use certificates that are not signed by a trusted authority, such as a self-signed certificate, on the mobile server. Set the following parameter in the `NETWORK` section on the client device:

```
[NETWORK]
DISABLE_SSL_CHECK=YES
```

This parameter enables the client to use the self-signed certificate for SSL encryption, but not to perform SSL authentication.

A.2.2.3 HTTP_PROXY

If user has a proxy between the mobile client and the mobile server, then in order for the Device Manager (dmagent) to access the mobile server to poll for command, then configure this parameter to the proxy server URL, including port number.

Format is `<hostname>:<port>`, as follows:

```
[NETWORK]
HTTP_PROXY=proxy.foo.com:8080
```

A.3 Sample OSE.INI and DEVMGR.INI Files

The following content is displayed from a sample `OSE.INI` file.

```
[SQLITE]
DATA_DIRECTORY=C:\mobileclient\sqlite
QUEUES=YES

[OSE]
RESUME=NO

[BGSYNC]
DISABLE=NO
```

The following content is displayed from a sample `DEVMGR.INI` file.

```
[NETWORK]
```

```
DISABLE_SSL_CHECK=YES
HTTP_PROXY=proxy.foo.com:8080
```

Index

A

B

C

D

E

F

H

I

L

M

N

O

P

Q

R

S

T

U