Oracle® Database Lite

Administration and Deployment Guide

10*g* (10.3.0)

B28922-01

April 2007

ORACLE®

Oracle Database Lite Administration and Deployment Guide 10*g* (10.3.0)

B28922-01

Contents

5 Managing Synchronization

6 Job Scheduler

7 Manage Your Devices

8 Manage Your Branch Office

9 Offline Instantiation

10 Using the Application Server OID With Mobile Server

11 Configure Security in Oracle Database Lite

F POLITE.INI Parameters

Preface

This preface introduces you to the *Oracle Database Lite Administration and Deployment 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, with good usability, to the disabled community. 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.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

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 1op2
  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 Managing the Mobile Server

Use the Mobile Workspace GUI tool for managing the Mobile Server; formally known as the Mobile Manager. In addition, Mobile clients on a Windows platform can use the Mobile Workspace GUI for managing Mobile clients with OC4J, Web-to-Go, BC4J, or Branch Office Mobile clients.

> **Note:** See Chapter 2, "Managing Your Mobile Client" for details on how to manage and synchronize each of the Mobile clients—including Linux, WinCE, and Win32—which are not managed using the Mobile Workspace GUI tool.

The Mobile Workspace only displays the relevant functionality depending on who logs in. Use the Mobile Manager to manage Mobile applications; use the Mobile Client Workspace to manage the application on the client-side device. The following sections detail how to use the Mobile Workspace for the Mobile Server. See Chapter 2, "Managing Your Mobile Client" for directions on how to use the Mobile Workspace for your Mobile Client.

- Section 1.1, "Using the Mobile Workspace"
- Section 1.2, "Using Mobile Manager to Manage Your Mobile Server"
- Section 1.3, "Enabling UIX Dynamic Image Generation on UNIX to See Mobile Manager Buttons"

1.1 Using the Mobile Workspace

The Mobile Workspace provides you access to your Mobile applications through hyperlinks in a Web browser. The following tabs are available when you use the Mobile Workspace for either managing the Mobile Server or the OC4J or Web-to-Go clients: Applications, Configuration, Help, Sync, and Log Off. See Section 2.3.1.1, "Instructions for Using the Mobile Workspace" for a description of the functions in each tab—especially the Configuration tab, as it enables you to configure options for your Workspace.

1.2 Using Mobile Manager to Manage Your Mobile Server

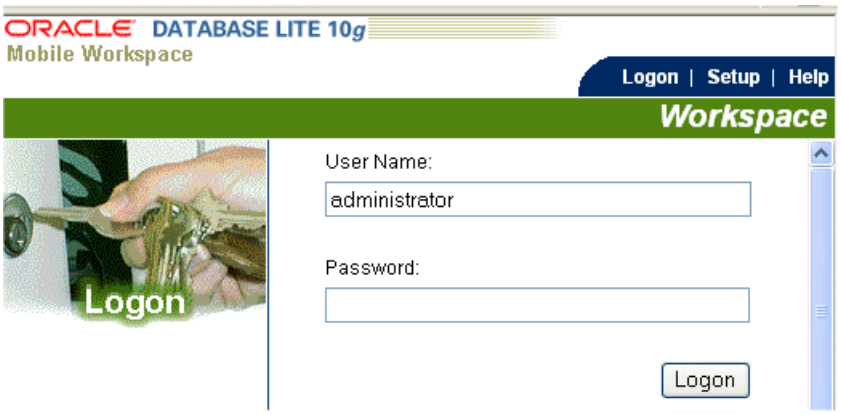
Mobile Manager is used to manage the Mobile Server. An administrator is the only user that is able to log on and use the Mobile Manager. To logon to the Mobile Manager, perform the following steps.

1. Using a browser, connect to the Mobile Server by entering the following URL.

```
http://<your_Mobile_Server_host_name>/webtogo
```

As Figure 1–1 displays, the Mobile Server Workspace appears with Mobile Server logon page.

Figure 1–1 Logon Page



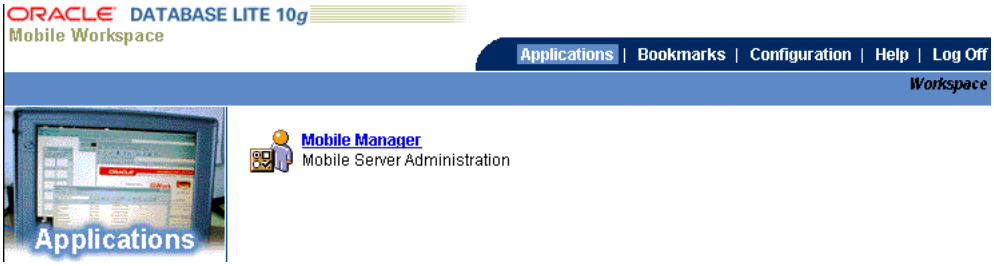
2. Log on to the Mobile Manager with the Mobile Server administrator username and password. A default administrator is created when you install with username/password of `administrator/admin`. Change the default password or create your own administrator user with appropriate username/password.

> **Note:** The username or password is limited to a maximum of 28 characters long. Also, the username cannot use multi-byte characters.

As shown in Figure 1–2, the Mobile Workspace displays the Mobile Manager in the Applications tab, which is the application available to administrators for managing any Mobile Server.

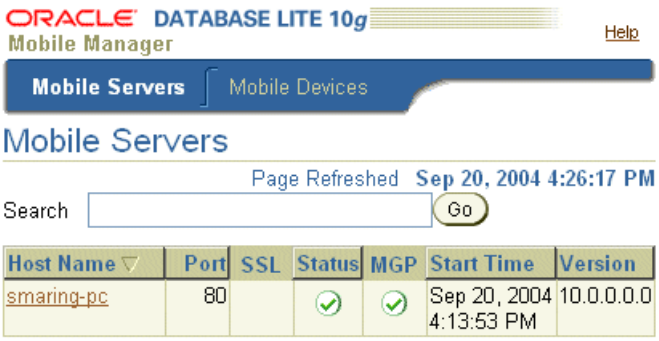
> **Note:** Section 1.1, "Using the Mobile Workspace" describes the functions of each of the tabs at the top of the Mobile Workspace.

Figure 1–2 Mobile Manager Workspace Page



3. Click the **Mobile Manager** icon or link. As Figure 1–3 displays, the Mobile Server farm page appears with a list of installed Mobile Servers that use this repository.

Figure 1–3 Mobile Server Farm Page



A Mobile Server farm is a group of Mobile Servers configured to run against the same repository. The Mobile Server farm page contains Mobile Server and Mobile Devices tabs. The following sections describe each component of the Mobile Server Farm. Topics include:

- Section 1.2.1, "Viewing Mobile Servers"
- Section 1.2.2, "Viewing Mobile Devices"

1.2.1 Viewing Mobile Servers

The Mobile Servers tab lists all Mobile Servers that are configured to run against the same repository. This page lists Mobile Server information such as Host name, Port, SSL enabled, Up or Down Status, the MGP instance, start time of the instance, and Mobile Server version (see Figure 1–3).

If a Mobile Server instance is not running (status column displays down), the hyperlink for the host name—where the Mobile Server exists—is not enabled. Refresh this page after a Mobile Server instance is started or stopped by clicking on Mobile Server link to see the updated status for the Mobile Server. Using search criteria based on host name, you can filter the Mobile Server display list for only those servers you are interested in.

The following sections describe the Mobile Manager components.

- Section 1.2.1.1, "Mobile Manager Home Page"
- Section 1.2.1.2, "Manage Applications"
- Section 1.2.1.3, "Manage Users"
- Section 1.2.1.4, "Mobile Server Administration"
- Section 1.2.1.5, "Data Synchronization"
- Section 1.2.1.6, "Job Scheduler"

1.2.1.1 Mobile Manager Home Page

The Mobile Manager home page displays the following Mobile Server information.

- General information such as current Mobile Server status, version, and mode.
- Database information such as database version, JDBC URL, JDBC Driver, JDBC version, and schema name.

- Data Synchronization information such as MGP Status, In Queue, Out Queue, and Error Queue details. See Chapter 5, "Managing Synchronization" for more information on Data Synchronization and the queues.
- Alert details that describe alert severity and the date and time on which the alert was triggered.
- Mobile Server components such as Data Synchronization and the Job Scheduler that enable you to schedule synchronization job sessions.

See Chapter 5, "Managing Synchronization" for more information on Data Synchronization. See Chapter 6, "Job Scheduler" for more information on the Job Scheduler.

1.2.1.2 Manage Applications

The Applications page enables the Mobile Server administrator to accomplish the following tasks.

1. Publish applications.
2. Create or edit application properties.
3. Resume, suspend, and delete applications.
4. Grant or revoke application access to users and groups.
5. Create or edit data subsetting parameters.
6. When required, provision Mobile application files for public use.
7. Add WAR files.

See Chapter 3, "Managing Your Mobile Applications" for more information on how to manage your applications.

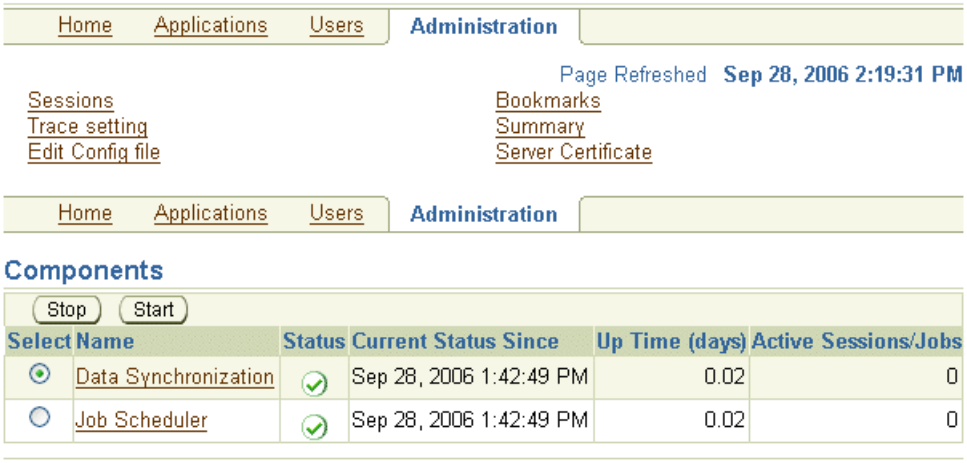
1.2.1.3 Manage Users

The Users page enables the Mobile Server administrator to manage groups and users and their permissions. See Chapter 4, "Managing Users and Groups" for full details.

1.2.1.4 Mobile Server Administration

Figure 1–4 shows the Administration page that enables the Mobile Server administrator to accomplish the following tasks:

Figure 1–4 Administration Page



1. View the sessions that are active. See Section 13.2, "Viewing Active User Sessions" for full details.
2. Edit trace settings. See Chapter 3, "Tracing and Logging" in the *Oracle Database Lite Troubleshooting and Tuning Guide* for full details.
3. Edit the configuration file. We prefer that you modify the configuration using the GUI tool; however, if you decide to edit the `webtogo.ora` file directly, you can access it with this link. See Appendix A, "Configuration Parameters for the WEBTOGO.ORA File" for details on the parameters.
4. Add bookmarks. See Chapter 14, "Adding Popular URLs as Bookmarks to Mobile Server Main Page" for full details.
5. View a summary of the database, JRE, and Operating System. See Section 13.1, "Viewing System Status Reports for the Server" for full details.
6. Upload an SSL certificate. When you are using an SSL connection with a reverse proxy in connection with a Web-to-Go client, you need to upload an SSL certificate. See Section 11.5.1.4, "Enable SSL When Using a Reverse Proxy" for full details.

1.2.1.5 Data Synchronization

When you select this link, you can configure and manage how synchronization occurs.

- Start or stop all synchronization activity for this Mobile Server.
- View the active sessions where synchronization is currently occuring.
- View statistics of previously executed synchronization sessions.
- Execute the Consperf performance tool to evaluate the synchronization performance.
- Modify parameters that affect how synchronization is performed.
- View the activity within the Repository—with each of the queues used for managing synchronization or with the users, publications, and publication items loaded in the repository.
- View all details about the MGP, including statistics, cycles and the Job Scheduler.

1.2.1.6 Job Scheduler

If you click on the Job Scheduler link, you can do the following:

- Start/stop the Job Scheduler
- View, enable, disable or delete any scheduled job.

1.2.2 Viewing Mobile Devices

The Mobile devices tab lists all Mobile devices that are registered with any Mobile Server, and are part of the same Mobile Server farm. The following sections briefly describe the functionality available to you in the Mobile Devices tab:

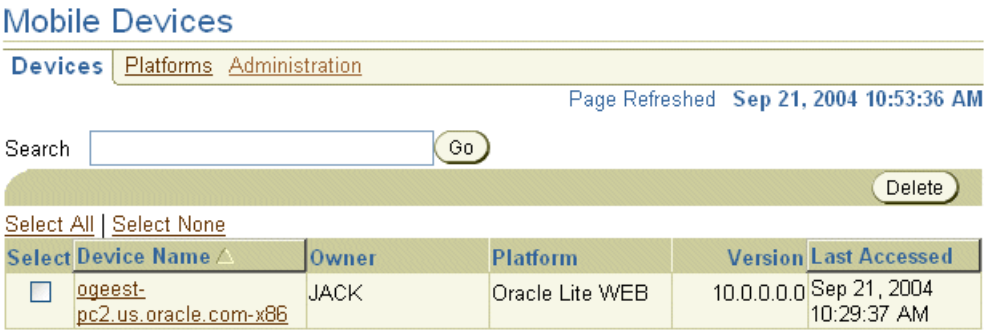
- Section 1.2.2.1, "Installed Mobile Devices"
- Section 1.2.2.2, "Mobile Device Platforms"

For full details on managing your Mobile devices, see Chapter 7, "Manage Your Devices".

1.2.2.1 Installed Mobile Devices

View the installed Mobile device information such as device name, owner, platform, version, and date and time on which it was last accessed. Figure 1–5 displays the Devices page.

Figure 1–5 Devices Page



To manage the Mobile device, click the Device Name link from the list. For full details on managing your Mobile devices, see Chapter 7, "Manage Your Devices".

1.2.2.2 Mobile Device Platforms

This page lists Mobile device platform information such as platform name, language, enabled, bootstrap, device count, and base platform.

There are hyperlinks that enable you to do the following:

- View device, installed Oracle Database Lite software, back-end database information, what is currently in the command queues, and the logs.
- Extend and manage device platforms—You can extend existing platforms for your own customization—adding other binaries to download or instructions on how to modify the client environment.
- Create commands to be sent to the Mobile device—You can create commands that execute on the device. These commands can start a synchronization, retrieve information, modify the client environment, and many other options.

For full details on managing your Mobile devices, see Chapter 7, "Manage Your Devices".

1.3 Enabling UIX Dynamic Image Generation on UNIX to See Mobile Manager Buttons

UIX generates images dynamically. On UNIX systems, this requires headless Java to be enabled or access to an X server to be enabled for the JVM. If you do not configure one of the following, then you will not see the buttons in the Mobile Manager.

- Section 1.3.1, "Headless Java"
- Section 1.3.2, "X Server Access"

1.3.1 Headless Java

Headless Java is only supported in Java 2 version 1.4 and later. In order to avoid X server configuration issues, enable headless operation by setting the Java option: `java.awt.headless` to `true`.

In Mobile Server standalone mode, set the parameter when you start the Mobile Server by modifying the `runmobileserver` script to include the following:

```
java -Djava.awt.headless=true -jar oc4j.jar
```

When deploying to an OC4J instance within the application server, the Java option must be specified within the `opmn.xml` file, as follows:

```
<oc4j instanceName="OC4J_Demos" gid="OC4J_Demos">
  <!-- OC4J configuration information here... -->
  <java-option value="-Djava.awt.headless=true "/>
</oc4j>
```

After modification, restart OC4J.

1.3.2 X Server Access

An accessible X server must be running at the same time as the Mobile Server. To make an X server accessible to the Mobile Server, the X server host grants access to the Mobile Server host through commands, such as `xhost +`. The Mobile Server host configures the `DISPLAY` environment variable to point to the X server, as follows:

```
set DISPLAY=<X server machine name>:<X server number>.<screen number>
```

In Mobile Server standalone mode, set the `DISPLAY` environment variable before starting the Mobile Server.

When deploying to an OC4J instance within the application server, the `DISPLAY` must be specified within the `opmn.xml` file, as follows:

```
<oc4j instanceName="OC4J_Demos" gid="OC4J_Demos">
  <!-- OC4J configuration information here... -->
  <environment>
    <prop name="DISPLAY" value="machinename:0.0"/>
  </environment>
</oc4j>
```

where `value` is `<machine name or IP address of the XServer>:<display number>`

After modification, restart OC4J.

2

Managing Your 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.

In general, the types of Mobile clients are as follows:

- Windows clients (such as Web-to-Go over OC4J, Web-to-Go, Branch Office, and BC4J): The application built for these clients uses a Java browser.
- Linux, Win32, and WinCE clients: These applications are client/server applications. Thus, start the application as you would start any application on these platforms.
- Broadbeam-enabled clients, which are used for network routing.

The following sections detail how to set up your Mobile client device and how to use the Oracle Database Lite technology on that device:

2.1 Start the Mobile Client

The following details how to start the Mobile client stack:

- Web clients (such as OC4J, Web-to-Go, Branch Office, and BC4J): The application built for these clients uses a Java browser. You can initiate the application or synchronization using the Mobile Workspace GUI.
- Linux, Win32, and WinCE clients: These applications are client/server applications. Thus, start the application as you would start any application on these platforms. You can initiate synchronization either by implementing it within your application using the synchronization APIs (see Chapter 3 "Synchronization" and Chapter 4, "Synchronization APIs" in the *Oracle Database Lite Developer's Guide* for more information) or by executing the `msync` executable, described in Section 2.3.2, "Use the mSync GUI to Initiate Synchronization of Your Linux, WinCE, and Win32 Client/Server Application Clients".
- Broadbeam-enabled clients, which are used for network routing, are described in Section 2.3.3, "Support Network Roaming for Devices With Broadbeam".

2.1.1 Start the Mobile Client

For Win32 and WinCE devices, you do not have to perform anything extra to start the Mobile client. However, for the Linux and Windows-based clients (OC4J and Web-to-Go), you may have to perform an extra step.

When you installed the Mobile client on Linux or Windows, it configured that the Mobile client should always be started automatically when the device is initiated. So, most of the time, you do not have to do anything to start the Mobile client. However, if you have a failure, you can manually start the Mobile client, as follows:

- Web Mobile client: If using the OC4J container for your Web applications, then start the Web OC4J Mobile client and its OC4J container by executing `runmobileclient`. If you want to use the Oracle Database Lite servlet container that was used prior to Release 10.3, then start the Web-to-Go Mobile client by executing the `webtogo` executable.

  Both executables are located in the `<mobile_client>/bin` directory.

- Linux Mobile client: Start the Linux Mobile client by executing the `webtogo.sh` file in the `<mobile_client>/bin`.

2.1.2 Log on to Mobile Client Workspace

If you are using a Windows-based client, then you must have installed the appropriate Windows Mobile client on your client machine. Upon machine startup, the Mobile client is automatically started. Then, you can connect to the Mobile Client Workspace with a browser by connecting to one of the following:

- `http://<client_machine>/webtogo`
- `http://localhost/webtogo`.

If the Mobile client is unexpectedly terminated, you can start the client by double-clicking on the following icon in the corner of your Windows desktop:

Or you can launch the Mobile client by accessing your Oracle Database Lite program group and choosing the Mobile client. Figure 2–1 displays the Mobile Client Workspace Logon page.

Figure 2–1 The Mobile Workspace Logon Page

Enter the username and password for the Mobile user and click Logon.

2.2 Synchronize or Execute Applications on the Mobile Client

The following details how to synchronize each Mobile client stack type:

- Web clients (such as OC4J, Web-to-Go, Branch Office, and BC4J): The application built for these clients uses a Java browser. Synchronization is either automatic or if you disable automatic synchronization, then you can synchronize through the Sync Tab in the Mobile Workspace GUI, as described in Section 2.3.1.1.5, "Sync Tab". You can initiate the application using the Mobile Workspace GUI, as described in Section 2.3.1.2, "Execute Mobile Applications Installed on Your Mobile Client".
- Linux, Win32, and WinCE clients: These applications are client/server applications. Thus, start the application as you would start any application on these platforms. You can initiate synchronization either by implementing it within your application using the synchronization APIs (see Chapter 3 "Synchronization" and Chapter 4, "Synchronization APIs" in the *Oracle Database Lite Developer's Guide* for more information) or by executing the `mSync` executable, described in Section 2.3.2, "Use the mSync GUI to Initiate Synchronization of Your Linux, WinCE, and Win32 Client/Server Application Clients".
- Broadbeam-enabled clients, which are used for network routing, are described in Section 2.3.3, "Support Network Roaming for Devices With Broadbeam".

> **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."`

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, mSQL, Web-to-Go or the client synchronization APIs.

The first synchronization for the Mobile client creates several Oracle Lite database (ODB) files on your client device for storing either Oracle Database Lite information or your application information. These ODB files are stored in the *<ORACLE_HOME>*`\Mobile\SDK\Oldb40` directory, as follows:

```
consroot.odb
webtogo.odb
<username>\acl.odb
          \conscli.odb
          \A<number>.odb
```

The following describes the purpose and access for these ODB files:

- The `consroot`, `acl`, and `conscli` ODB files are used for Oracle Database Lite data. The `webtogo.odb` file is only present when using a Web-to-Go application. If you want to access the information within these ODB files, use `SYSTEM/MANAGER` as the username/password.
- The snapshot data for the application is contained within the `A<number>.odb` file, where `<number>` is a randomly-generated number. You can specify the name of this ODB file when you create your publication; see Chapter 5, "Using Mobile

Database Workbench to Create Publications" in the *Oracle Database Lite Developer's Guide* for details.

If you want to access the snapshot data directly using `msql` or any other SQL tool, use `SYSTEM` as the username and provide the password for the user that owns the snapshot. For example, if the user `John/Foo` initiated the synchronization, to access the `A<number>.odb` file for this snapshot, you would use `SYSTEM/Foo` as the username/password.

2.3 Manage the Mobile Client

There are several tools that you can use on the client to manage functionality. The following sections describe other tools that you can use in each platform.

- Section 2.3.1, "Use the Mobile Client Workspace to Manage Your Web Clients"
- Section 2.3.2, "Use the mSync GUI to Initiate Synchronization of Your Linux, WinCE, and Win32 Client/Server Application Clients"
- Section 2.3.3, "Support Network Roaming for Devices With Broadbeam"
- Section 2.3.4, "Reset the Mobile User Password"
- Section 2.3.5, "Use the Device Manager Client GUI to Manage the Client-Side Device"
- Section 2.3.6, "Initiate Updates of Oracle Database Lite Software from the Client"
- Section 2.3.7, "Defragmentation and Reducing Size of the Client Application Databases"
- Section 2.3.8, "Communicate Between the Internet and Intranet Through a Reverse Proxy"

2.3.1 Use the Mobile Client Workspace to Manage Your Web Clients

The Mobile Workspace GUI tool is used for both the Mobile Manager (of the Mobile Server) and for managing OC4J, Web-to-Go, BC4J, or Branch Office Mobile clients. The Mobile Workspace only displays the relevant functionality for the user that logs in. Use the Mobile Manager to manage Mobile applications; use the Mobile Client Workspace to manage the application on the client-side device.

The following sections detail how to use the Mobile Workspace for your Web clients and how to synchronize for each type of client:

Note: See Chapter 1, "Managing the Mobile Server" for information on how to use the Mobile Workspace for managing the Mobile Server.

- Section 2.3.1.1, "Instructions for Using the Mobile Workspace"
- Section 2.3.1.2, "Execute Mobile Applications Installed on Your Mobile Client"
- Section 2.3.1.3, "Customize the Mobile Client Workspace"
- Section 2.3.1.4, "Schedule Data Synchronization Jobs"

2.3.1.1 Instructions for Using the Mobile Workspace

The Mobile Workspace provides you access to your Mobile applications through hyperlinks in a Web browser. The following tabs are available when you use the Mobile Workspace for a Web client:

2.3.1.1.1 Applications Tab The Applications tab displays the list of applications that have been installed on the client, which you can execute. To access a Mobile application, click the icon or application name. The Mobile Workspace allows you to execute multiple applications concurrently in separate browsers.

> **Note:** When you log on as an Administrator, the Mobile Manager is listed as the available application.

2.3.1.1.2 Configuration Tab To modify your Workspace configuration, select the Configuration tab.

> **Note:** When you select the Configuration tab, maximize the window to your display. If you do not, you might not be able to see all of the configuration options on the left.

The following options appear on the left:

- Workspace Settings: Configure your Mobile Workspace settings, such as display options and Web client options for OC4J or Web-to-Go clients. Table 2–1 discusses the Web client settings:

Table 2–1 Workspace Configuration Options

Label	Description
Display icons?	Enables you to display Web application icons. To display Web application icons, select **Yes**. If you do not want to display Web application icons, select **No**.
Display description?	Enables you to display Web application descriptions. To display Web application descriptions, select **Yes**. If you do not want to display Web application descriptions, select **No**.
Applications per row	Enables you to specify the number of applications arranged in a horizontal row in the Workspace.
Use default settings for synchronization?	If you select this option, all applications (including new applications) are synchronized at once. If you de-select this option, then you can choose which applications synchronize. Using this list, you can reduce sychronization time by selecting which applications you want to synchronize. In this mode, you are notified when new applications become available.
Mobile Client for OC4J or Web-to-Go Mode	Selecting **Always Offline** enables you to work continuously. If you want to use the backwards compatible option of offline/online, select the **Online/Offline** option.

Table 2–1 (Cont.) Workspace Configuration Options

Label	Description
Ask before ugrading the Mobile Client for OC4J or Web-to-Go?	Selecting this option instructs the Mobile client to ask you before downloading newer software versions for the Mobile client. If you do not select this option, your Mobile client is automatically upgraded the next time you synchronize if a new version is available.
Enable Automatic Sync?	By default, this is enabled. If checked, then automatic sync is automatically enabled. Uncheck if you want to manually synchronize this client yourself.
Number of Automatic Sync History Messages to be displayed per page.	This defines the number of history records to display on the Automatic Sync History page. By default, this value is 25.

- Application Settings: Select the applications that you want to synchronize by default. In the default synchronization mode, the Mobile Server synchronizes your client applications automatically with the Mobile Server. See "Configure Application Synchronization Default" for full details.
- Change Password: The password is stored on both the client and the Mobile server. To ensure that the password is modified on both the client and the Mobile server, only change the password using the Client Workspace when you have a connection to the Mobile Server. See Section 2.3.4, "Reset the Mobile User Password" for more details.
- Create a Data Synchronization Job: You can schedule a time when synchronization is to be initiated on the client. This is configured as a job initiated by the client. There is a small job engine on the client, so when you set up a job to execute at a certain time and interval, it initiates the application at the specified time. The only job that you can schedule on the client is for synchronization. You should not schedule multiple client synchronization jobs for the same time on the client. In fact, you should make sure that all other connections to the database, such as msql, jdbc, and so on, are closed before starting the synchronization. For more information on scheduling jobs on the server-side, see Chapter 6, "Job Scheduler".

  Scheduled replication jobs will not run on client if there are any pending transactions that have not been committed on that client.
- In addition, you can disable or enable remote access for the Mobile client. See "Enable or Disable Remote Access for Mobile Client" for more information.

Configure Application Synchronization Default

The Application Settings option allows you to select the applications that you want to assign to the default synchronization setting. In the default mode, the Mobile client synchronizes your client application with the Mobile Server when the user requests synchronization. Selecting fewer applications decreases the amount of data to download and speeds up the synchronization process.

To select the applications for synchronization, click **Application Settings**. The application settings page appears in the right frame of the Mobile Workspace and displays the following information:

Table 2–2 describes the Application Settings page.

Table 2–2 Application Settings Page Description

Label	Description
Synchronize	If selected, the Mobile client synchronizes your client application with the Mobile Server when the user initiates synchronization.

Select the Synchronize check box next to the application and click **Save**. If you make an error, click **Reset** to return to the previous settings.

Enable or Disable Remote Access for Mobile Client

To block a remote machine from getting access to the Mobile Client for OC4J, Web-to-Go, or BC4J, set the `DISABLE_REMOTE_ACCESS` parameter in the client-side `webtogo.ora` file to `YES`. Once this parameter is set to `YES` and the Mobile client is restarted, only the request coming from the local machine is served by the Mobile client listener. Any other request is blocked and not served.

For Branch Office clients, this parameter must be set to `NO`, as all clients must have remote access. For more information on the `DISABLE_REMOTE_ACCESS` parameter, see Section A.2, "[WEBTOGO]".

2.3.1.1.3 Automatic Sync History This screen shows you the history for all automatic synchronization events that occurred for your Mobile client. The following information is displayed about each event:

Table 2–3

Label	Description
Event code	An internal code that supplies more information to Oracle Support, if called.
Event type	An icon showing the outcome of the synchronization, as follows: ■ An `X` demonstrates there was an error. ■ An exclamation point indicates a warning. ■ An `i` indicates the synchronization was successful.
Timestamp	Date and time of the automatic synchronization.
Event Description	A message that describes the outcome of the automatic synchronization between your Mobile client and the back-end Oracle database.
Delete	Check the delete checkbox and click **Save** if you no longer wish to see the history for this event.

In addition, you can perform the following:

- If you have more than a single page of events listed, click Previous and Next to view all events.
- To delete one or more events, select the Delete checkbox of these items and then click **Save**.
- To delete all events on this screen, click **Select All** and then click **Save**.
- To deselect all Delete checkboxes, click **Clear All**.

2.3.1.1.4 Help Tab The Help tab launches the online help system.

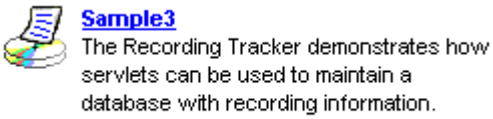
2.3.1.1.5 Sync Tab If your device is not currently connected, then you can synchronize data and applications at any time through the Sync tab. When you select the Sync tab, a synchronization of your data is initiated with the Oracle database. In addition, application updates and new applications accessible to the user are downloaded from the Mobile Server. After synchronization, you can work continuously disconnected from the Mobile Server. You only need a connection to the Mobile Server when you synchronize your data and applications.

For information on setting synchronization options, see Section 2.3.1.1.2, "Configuration Tab".

2.3.1.1.6 Log Off Tab The Log Off tab automatically closes all running applications and returns you to the logon page.

2.3.1.2 Execute Mobile Applications Installed on Your Mobile Client

Web-to-Go and OC4J applications appear in the Mobile Workspace with an icon, name, and description.



The icon and application name are both hyperlinks. To execute a Web-to-Go or OC4J application, click either the icon or application name.

The Web Mobile clients, such as OC4J or Web-to-Go, enable you to work disconnected from the Mobile Server. You need a connection to the Mobile Server only when you choose to synchronize any data changes from the client with the Oracle database.

The Web Mobile client propagates the tables that your applications use on the Mobile Server to the Mobile client as database snapshots. When you define your snapshot, you can use the `SQL WHERE` clause to specify a parameterized SQL query, where only the row data that your application uses is downloaded to the client. Thus, you can define what is downloaded to the client: the entire contents of the table or the subset of information that is relevant to the specific user. Most applications specify a particular subset of data that is relevant only to the user to be downloaded.

You can work continuously with Web Mobile clients storing your data changes in the Oracle Lite database. When you click the Sync tab, the Web Mobile client updates the Oracle database with any data changes you made on your client. The Mobile Server downloads any new applications, application changes, or data changes to your client.

2.3.1.3 Customize the Mobile Client Workspace

You can customize the Mobile Client Workspace. See Section 6.2.9 "Customizing the Workspace Application" in the *Oracle Database Lite Developer's Guide*.

2.3.1.4 Schedule Data Synchronization Jobs

The Mobile Workspace enables you to create data synchronization jobs for your site from the OC4J or Web-to-Go Mobile client. This synchronization job automatically triggers synchronization with the Mobile Server at the start date and at the specified time for this job that you set using the Mobile Workspace. See Section 2.3.1.1.2, "Configuration Tab" for more information.

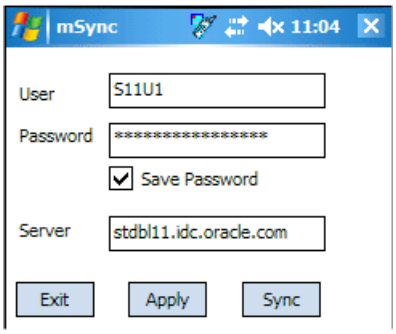
2.3.2 Use the mSync GUI to Initiate Synchronization of Your Linux, WinCE, and Win32 Client/Server Application Clients

The following details how to synchronize the different Mobile client platforms:

- Web clients (such as OC4J, Web-to-Go, Branch Office, and BC4J): The application built for these clients uses a Java browser. You can initiate the application or synchronization using the Mobile Workspace GUI.
- Linux, Win32, and WinCE clients: These applications are client/server applications. Thus, start the application as you would start any application on these platforms. You can initiate synchronization either by implementing it within your application using the synchronization APIs (see Chapter 3 "Synchronization" and Chapter 4, "Synchronization APIs" in the *Oracle Database Lite Developer's Guide* for more information) or by executing the `mSync` executable, described in Section 2.3.2.1, "Use Mobile Client Tools on Linux".
- Broadbeam-enabled clients, which are used for network routing, are described in Section 2.3.3, "Support Network Roaming for Devices With Broadbeam".

You can initiate synchronization of the client using the mSync GUI, as shown in Figure 2–2.

Figure 2–2 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 `/bin` subdirectory under the directory where you installed the Mobile client. Modify the following supplied values, if incorrect:

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

> **Note:** The username and password are limited to a maximum of 28 characters long.

- 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.

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

Figure 2–3 The mSync Tools Selection

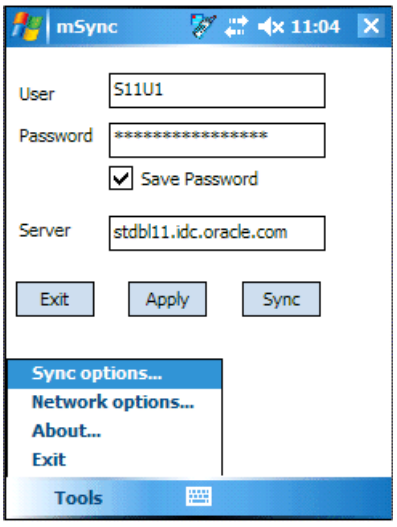


Figure 2–4 displays the Network options screen where you can specify a proxy if your network provider requires that you use a proxy server to access the internet. . Click **Use Proxy** to use a proxy and then enter the proxy server and port number.

Figure 2–4 The mSync Network Options Selection

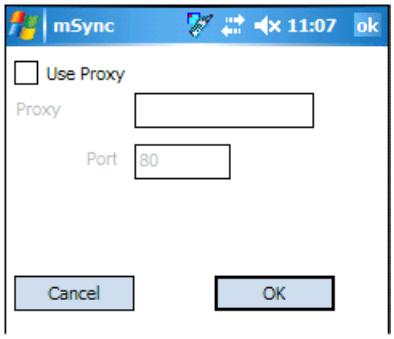
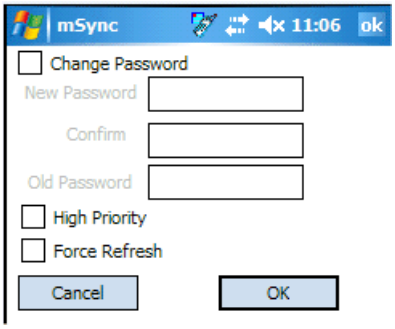


Figure 2–5 displays the Sync Options screen where you can specify the following:

- 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 2.3.4, "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.
- 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 2–5 The mSync Options Selection



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 2.3.6, "Initiate Updates of Oracle Database Lite Software from the Client" for more information.

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

2.3.2.1 Use Mobile Client Tools on Linux

The 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.

2.3.3 Support Network Roaming for Devices With Broadbeam

Normally, Oracle Database Lite associates devices with a single network provider for Device Manager. The Mobile Server sends commands to the device based on the registered network provider, which works seamlessly as long as the device does not roam over multiple network providers.

To support network roaming with device manager commands, you can integrate Oracle Lite with Broadbeam ExpressQ (Release 4.1 SP3) and Smart IP (Release 3.1 SP1). Broadbeam enables sending commands across any network type that is supported by Broadbeam products. Device Manager Commands use the Broadbeam messaging engine. Mobile Client may synchronize data over Smart IP.

2.3.4 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 either mSync UI or Client Workspace. Only modify the password using these tools 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.

See Section 11.2, "Which Password is Which?" for details on all Oracle Lite Database passwords.

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

On any client, you can manage the Mobile device client software using the Oracle Lite Device Manager. See Section 7.9, "Using the Device Manager Agent (dmagent) on the Client" for a full description.

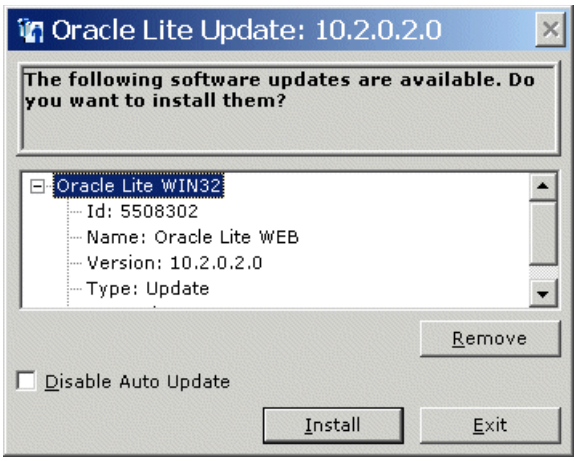
2.3.6 Initiate Updates of Oracle Database Lite Software from the Client

On Windows clients, you can initiate a request for software updates from the Mobile Server by executing the Oracle Lite Update tool, as shown in Figure 2–6. To execute, choose **Oracle Lite Update** from the Oracle Database Lite Programs list or enter `update` on the command line.

For each type of client, the update tool acts as follows:

- The mSync tool automatically launches this tool if and only if a software update is available.
- For Web-to-Go clients, the client prompts the user in a Web page if an update is available.
- For Branch Office clients and applications that use the synchronization APIs, this tool is not automatically executed. Instead, if you want to launch the update tool to check for software updates, then explicitly enter `update` on the command line.

Figure 2–6 Updating Oracle Database Lite Software



When updates are located, you can select items that you do not want to update and click **Remove**. When all updates are satisfactory, click **Install**. When you are finished, click **Exit**.

If you check the Disable Auto Update checkbox, then the next time you execute mSync, this tool is not automatically executed. You can also disable automatic updates from the Mobile Manager. See Section 7.5.3, "Allowing Software Upgrades to All Mobile Devices in a Platform" for more information.

2.3.7 Defragmentation and Reducing Size of the Client Application Databases

On each client device, an Oracle Lite database stores the application data—either as an embedded database that exists solely for the application or as a repository for data for a specific user that is synchronized with a back-end Oracle database.

You can use the DefragDB utility on your database to perform the following optimizations:

- Reduce size of Oracle Lite databases by defragmenting the Oracle Lite database.
- Remove any BLOB data from the Oracle Lite database. Prior to release 10.3, all data was stored in an `.odb` file that had a 4 GB limit. This was restrictive to storing BLOB data, such as media files. From release 10.3 onwards, all BLOB data—both binary and character— and indexes are stored in separate files with the extension of `.obs` for Oracle Blob Store. This changes the size limit on your device to the operating system file size limitations or 16 terabytes.

> **Note:** This tool removes any Blob data currently in your Oracle Lite database and stores it in its own `.obs` file. However, if you do not run this tool, you can continue to work seamlessly. Any new Blob data is stored in an `.obs` file; the pre-existing Blob data can continue to reside in the `.odb` file.

Use the DefragDB tool to defragment Oracle Lite databases, which reduces their size by compacting them and removing any Blob data from within the database into its own `.obs` file. The DefragDB tool is a UI dialog which allows you to choose which databases to defragment. This tool defragments databases by dumping each database into a file and then reloading it from this file. Alternatively, you can use the

command-line interface: `olmig.exe`. Both tools exist in the *<ORACLE_HOME>*`/Mobile/Sdk/bin` on your desktop or in `\OraCE` on a WinCE device.

> **Note:** Currently the tool runs on Win32 desktop and Windows CE devices.

The following sections describe this tool:

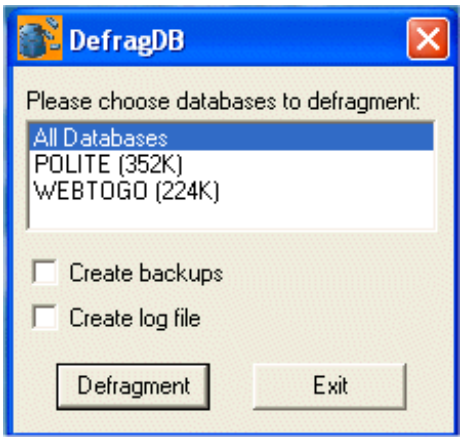
- Section 2.3.7.1, "Execute DefragDB"
- Section 2.3.7.2, "Pause or Cancel Defragmentation"
- Section 2.3.7.3, "Execute DefragDB With Command-Line"

2.3.7.1 Execute DefragDB

To start the tool, either double-click on the Oracle Database Lite Degramentation icon or execute `DefragDB.exe`, which brings up the following screen:

You can execute the Oracle Database Lite Defragmentation tool on the client for all applications. Before executing this tool, you must stop all applications, as the database is erased during this process. This includes the Oracle Lite applications, such as the Sync Agent, Web-to-Go, and so on. To stop the Sync Agent, see Section 5.4.2, "Start, Stop, or Get Status for Automatic Synchronization".

Figure 2–7 DefragDB GUI



All application databases are listed on this screen. Select the existing databases on your PC (or WinCE device) on which you want to perfrom the defragmentation.

- Click **Defragment** to defragment all databases. This tool performs a defragmentation on one database at a time.

> **Note:** In the worse case scenario, the defragmentation process requires three times the space of the database to complete the process. Thus, if you do not have enough space to defragment your larger databases, you will receive a warning notice about the database that is too large to complete the process. In order to continue, either free up enough space to enable the process to complete or return to the main screen and select all databases except for the offending one.

- To defragment specific databases, select the databases desired from the list and click **Defragment**.

> **Note:** To cancel out without performing any defragmentation, click **Cancel**. See Section 2.3.7.2, "Pause or Cancel Defragmentation" for more information.

In addition, select the following checkboxes, as appropriate:

- Create backup: provides a backup of the original copy of the database before defragmentation. The backup copy has the same name with a `.bak` extension. For example, `C:\orant\oldb40\polite.odb` becomes `C:\orant\oldb40\polite.odb.bak`. Thus, you can restore the database if an error occurs during defragmentation. In addition, the blob storage file (`.obs`) is backed up. To restore to the original version, rename these files back to the original names without the `.bak` extensions.
- Create log: provides a log, `defragdb.log`, of the defragmentation process, which is useful for the developer in diagnosing any problems that may occur during the defragmentation.

When you click **Defragment**, the process initiates, which brings up a window that displays messages about the progress of the defragmentation. This same log can be saved in the `defradb.log` file. See the status bar at the bottom for the final status: defragmenting, success, or fail.

If your database is encrypted, another dialog prompts for the user name and password for the encrypted database. Select the user name from the list of users and enter the password for that user. You only need to enter the password once if all databases are encrypted with the same password.

There are 2 files created during the defrag which should be deleted automatically by DefragDB when it finishes:

- dump file, which has the same name as the database, but with a `.dmp` extension.
- the newly loaded database, which has the same name as the database, but witha `_defrag` suffix. This file is renamed to the database name once the load completes.

2.3.7.2 Pause or Cancel Defragmentation

If you are defragmenting large databases, this may take some time. For example, it takes about 5 minutes on a desktop to defragment a 200 MB database; however, on WinCE devices, the defragmentation performs slower than the desktop machine. The amount of time this process takes is proportional to the size of the database. Thus, if you need to pause or cancel this process by clicking the **Cancel** button. This pauses the process. To continue the defragmentation, select **No** on the Cancel/Continue prompt. Select Yes on this prompt to stop the process entirely. If you cancel, the database remains in the original state, so your applications still perform normally.

2.3.7.3 Execute DefragDB With Command-Line

You can use automatic defragmentation, execute defragmentation within your application, or use additional options only available with the command-line tool. The following shows two examples: the first deframents all databases, the second defragments a specific database identified by name.

```
olmig -defrag all
or
```

```
olmig -defrag dbname
```

The usage for `olmig.exe` is as follows:

```
olmig -dump|-load|-defrag|-restore <dbName>|all [options]
```

Where:

- `-dump` : dump the database to a dump file (default `dbName.odb.dmp`). You can separate the functionality of the defragmentation into the dumping and loading. You can perform these functions at separate times, if desired.
- `-load` : load database from the dump file. This completes the defragmentation process and should only be executed after a dump is performed.
- `-defrag` : defragment database, which performs both the dump and load for the database.
- `-restore` : restore a database from a backup. If an error occurred, restore the saved original database.
- `<dbName>` or `all` : perform the actions on a specfic database or on all Oracle Lite databases on the machine or device.

Options include:

- `-auto` : exit the dialog when upgrade is done. When you invoke the command-line, a GUI is initiated. If you want this screen to exit when finished, provide the `-auto` option.
- `-backup` : backup the database to `dbName.odb.bak`
- `-readonly` : connect to a database that is read-only. During a dump only, you can dump a read-only database, such as one that may exist on a CD-ROM. Since the dump normally is written to the same location as the database, you must also provide the directory location and filename for the output. Thus, this option is only valid if used in combination with the `-dump` and `-file` options.
- `-nosingle` : do not enter single-user mode. Normally, only a single user can connect to the database while performing a dump, load, or defragmentation. That way, other users are not allowed to update an Oracle Lite database that is currently in the middle of a defragmentation, dump, or load activity. However, if you are performing a dump, you can use this option to enable other users to continue to execute their applications against the database. This cannot be allowed during any load activity.
- `-log logfile` : append messages to the specified file
- `-file dumpfile` : use the specified dump file, instead of the default file
- `-dot interval` : print a dot after processing the number of objects designated by `interval`; set the `interval` to 0 to disable this option.
- `-commit interval` : during the load process, this designates the number of rows after which to perform a commit; set the `interval` to 0 for no commit.
- `-passwd passwd` : specify a connect password for encrypted databases

2.3.8 Communicate Between the Internet and Intranet Through a Reverse Proxy

If your Mobile client is on either side of the firewall, you can set up a proxy or reverse proxy to facilitate communication between the Mobile client and Mobile Server. See Section 11.5, "Using a Firewall Proxy or Reverse Proxy".

3

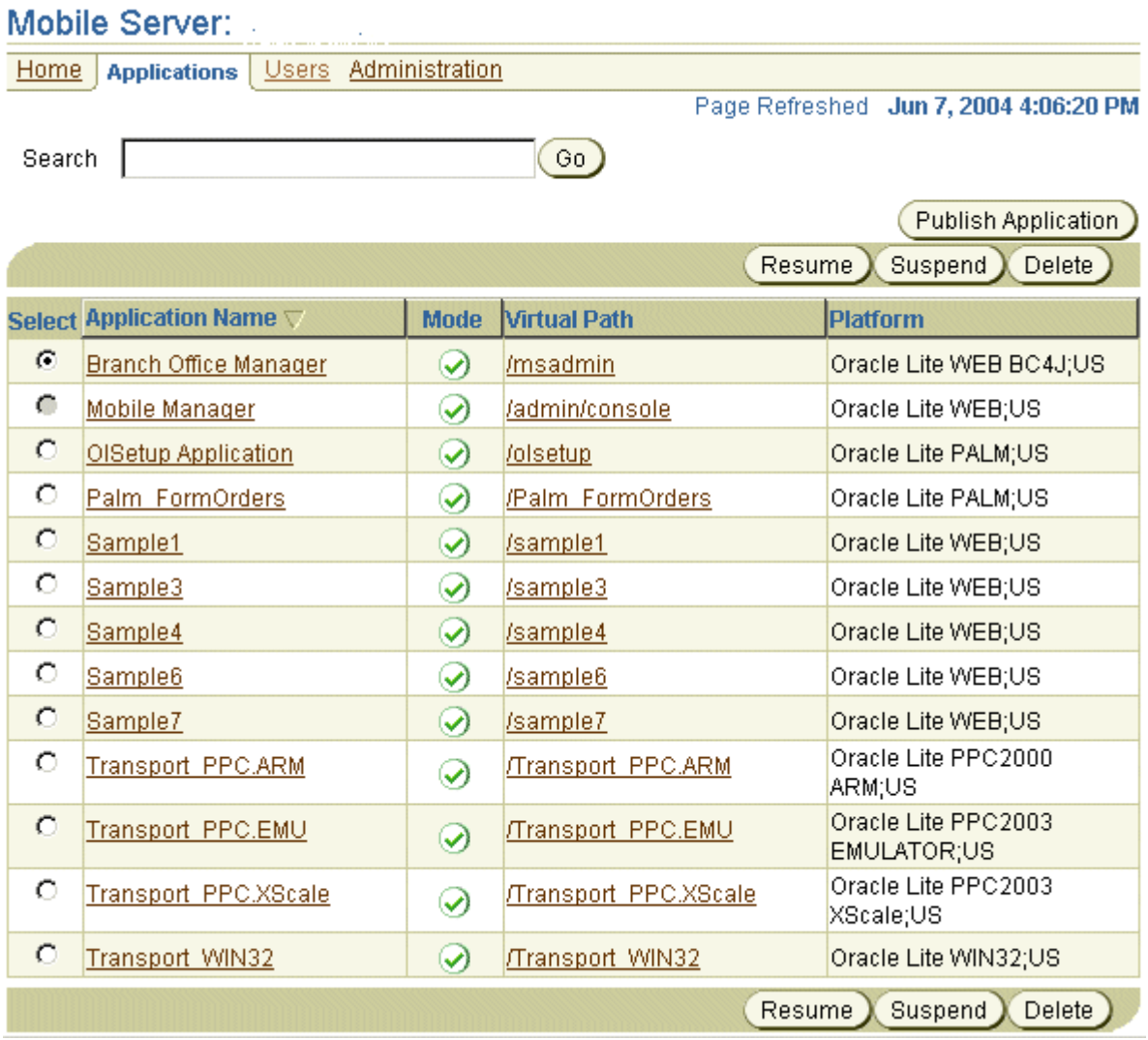
Managing Your Mobile Applications

The administrator manages applications through the following tasks:

3.1 Listing Applications

You can view all applications that are currently published on this Mobile Server from the Mobile Server home page. Click **Applications**. Figure 3–1 displays the Applications page, which lists existing applications and corresponding virtual paths.

Figure 3–1 Applications Page



To search applications, enter your application name in the **Application Name** field and click **Go**. The Applications page displays the search result under the Application Name column.

3.2 Publishing Applications to the Mobile Server Repository

A developer builds the Mobile application and packages it together with a publication. At this point, the application is ready to be published to the Mobile Server through one of the following options:

Note: If you developed the application on a machine remote to where the Mobile Server is installed, copy the application WAR or JAR file to this machine.

- An organizer or administrator can publish the application using the packaging wizard.

- An administrator can publish the application using the Mobile Manager from the Applications page with the Publish Application button, as shown in Figure 3–1.

3.3 Deleting an Application

To delete any application, select the application on the applications page (see Figure 3–1) and click **Delete**. This deletes it from the Mobile Server repository. However, because of the way Mobile Server is designed, you may wish to simply republish the application and not to delete an application.

By deleting the application, all synchronization related objects—such as publication, publication items, sequences and script definitions—are removed from the Mobile Server repository. However, the actual application tables in the back-end Oracle database are not removed.

Once the application has been deleted, existing Mobile clients can no longer synchronize for this application—ever again. Even if you were to publish the same application again, synchronizing existing Mobile clients will still fail. Instead, all transactions uploaded by the Mobile clients are placed in the error queue and require manual intervention from the administrator.

When a publication item is created, the Mobile Server assignes a unique identifier to each publication item. These unique identifiers are then used during the synchronization process to map snapshot tables on the Mobile clients to publication items and base tables on the back-end server. When a Mobile client uploads data, it uses this unique identifier to inform the Mobile Server as to which publication item this data is designated. If an application is deleted and republished, the Mobile Server uses a different set of identifiers for the publication items than the Mobile client; thus, the Mobile client and the publication item use different identifiers and the synchronization fails.

You should never delete an application if you have existing Mobile clients that still need to synchronize with the Mobile Server. If you want to modify an existing application, do not delete the application using Mobile Manager.

Instead, simply republish the application and indicate that you wish to overwrite the existing application.

3.4 Manage Application and Connection Properties

From the Applications page, you can modify application and connection properties for each application. Click on the application name to bring up its Properties page, shown in Figure 3–2.

Figure 3–2 Application Properties Page

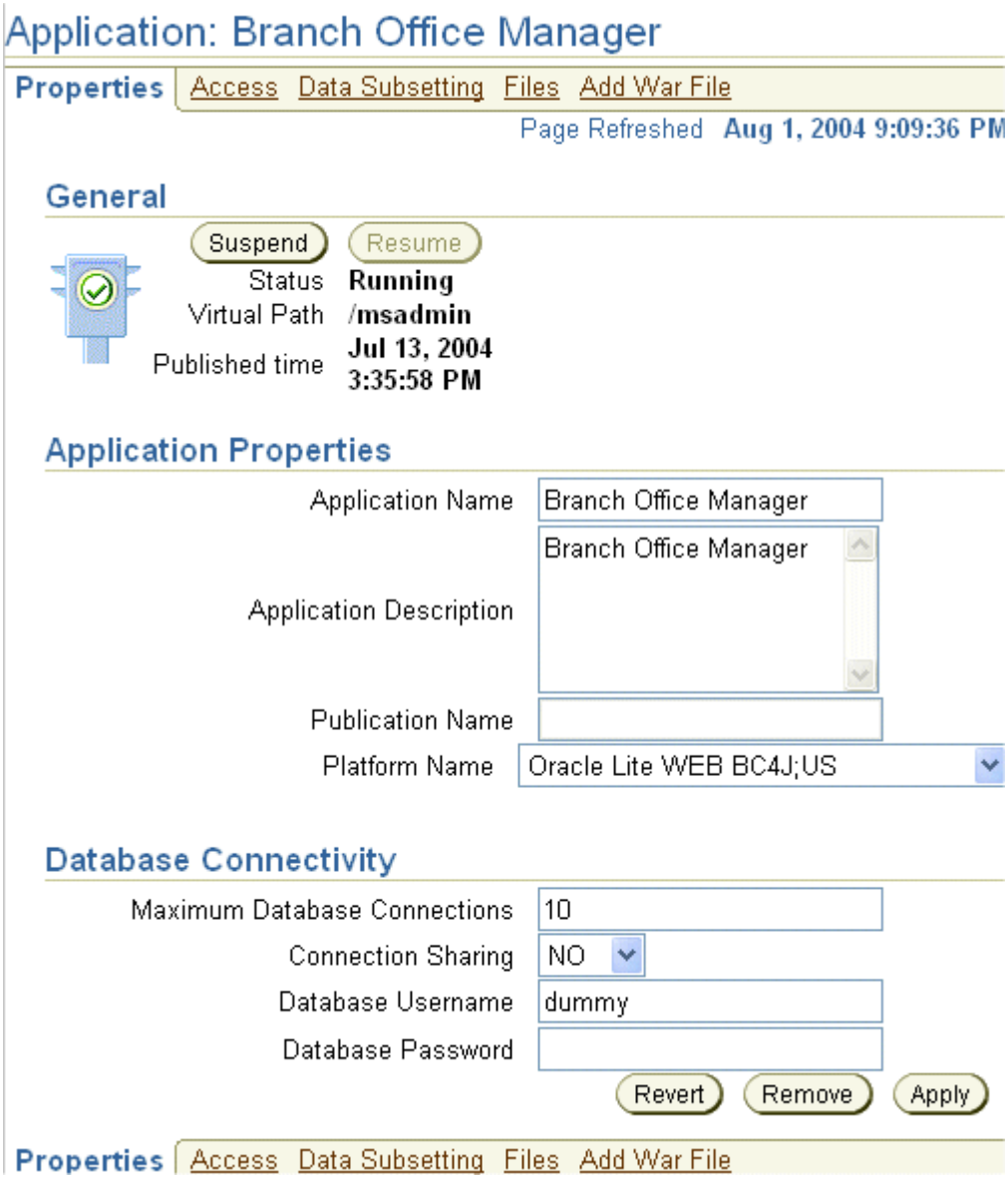


Table 3–1 describes the application properties that you can modify in this screen.

Table 3–1 Application Properties Page Description

Field	Description
Application Name	Name of your Mobile application.
Application Description	A brief description of your Mobile application.
Publication Name	Your application is published with a publication that contains the definition of the snapshot data for the clients. This field displays the publication name of the Mobile application. You cannot modify this field.
Platform Name	The platform name consists of the platform type and the language of the application. You can modify this platform to another type as displayed in the pull-down list.

Table 3–2 describes the following data connectivity properties that are available for OC4J or Web-to-Go applications.

- You can limit the number of connections are allowed to the database. To manage the performance and available resources of the database, you may want to set a limit of how many connections each application can have open at any given time.
- You can enable a connection pool for your OC4J or Web-to-Go application to use. Connection pools are set up for performance reasons. As each connection request comes in, a connection from the pool is used for the incoming request. When the request ends, the connection is returned to the pool. This eliminates the time necessary for creating and destroying the connections each time a new request comes in.

Table 3–2 Data Connectivity Properties

Property	Descripton
Maximum Database Connections	Number of maximum database connections used by your Mobile application.
Connection Sharing	Select **Yes** if you want to use connection pooling.
Database User Name	Username for the schema used by the application in the database, which is limited to 28 characters.
Database Password	Password of the schema user, which is limited to 28 characters.

To retain the modified application properties, click **Apply**. To remove the application, click **Remove**. To reset the Application Properties page, click **Revert**.

Alternatively, you can configure connection parameters in the `CONSOLIDATOR` section in the `webtogo.ora` file, as follows:

- `MAX_CONNECTIONS`—specifies the maximum number of JDBC connections that can be open at one time by the Mobile Server.
- `CONNECTION_TIMEOUT`—specifies the JDBC connection timeout for the synchronization session.

For full details, see Section A.7, "[CONSOLIDATOR]".

3.5 Managing User-Specific Application Parameters (Data Subsetting)

In retrieving data for each user, the application often requires that a parameter is set defining the type of data to retrieve. Set this parameter, also known as data subsetting, in one of two places: on the Data Subsetting page off the Applications page or on the Data Subsetting page off the Users page. See Section 4.3, "Managing Application Parameter Input (Data Subsetting)" for directions on how to manage the input parameter values for the application from the User page.

What is Data Subsetting? When you set up your publication item, you may have set up an input parameter that defines what snapshot of data is to be retrieved for this user. For example, if you have an application that retrieves the customer base for each sales manager, the application needs to know the sales manager's identification number to retrieve the data specific to each manager. Thus, if you set up each sales manager as a unique user and set their identification number in the data subsetting screen, then the application is provided that unique information for retrieving data.

1. Navigate to the Applications page. Click the specific application.

2. Click **Data Subsetting**. The Data Subsetting page enables the administrator to add parameter input for each user of this application. This displays all of the users that the application is associated with.
3. Select the user for which you want to add the parameter value.
4. Enter the parameter values for the application.
5. Click **Save**.

3.6 Managing Access Privileges for Users and Groups

Similar to Data Subsetting, you can set the access privileges for the application either from the Users page or from the Applications page—except for groups. Groups can only be given access to applications from the Applications page. See Section 4.2.1, "Grant or Revoke Application Access to Users" for directions on how to manage the access privileges from the user page.

The Mobile Server Administrator grants access privileges to Mobile applications by designating the users that can access these applications. This section enables an administrator to grant or revoke application access to users and groups. Topics include:

- Section 3.6.1, "Grant Application Access to Users and Groups"
- Section 3.6.2, "Revoke Application Access to Users and Groups"

3.6.1 Grant Application Access to Users and Groups

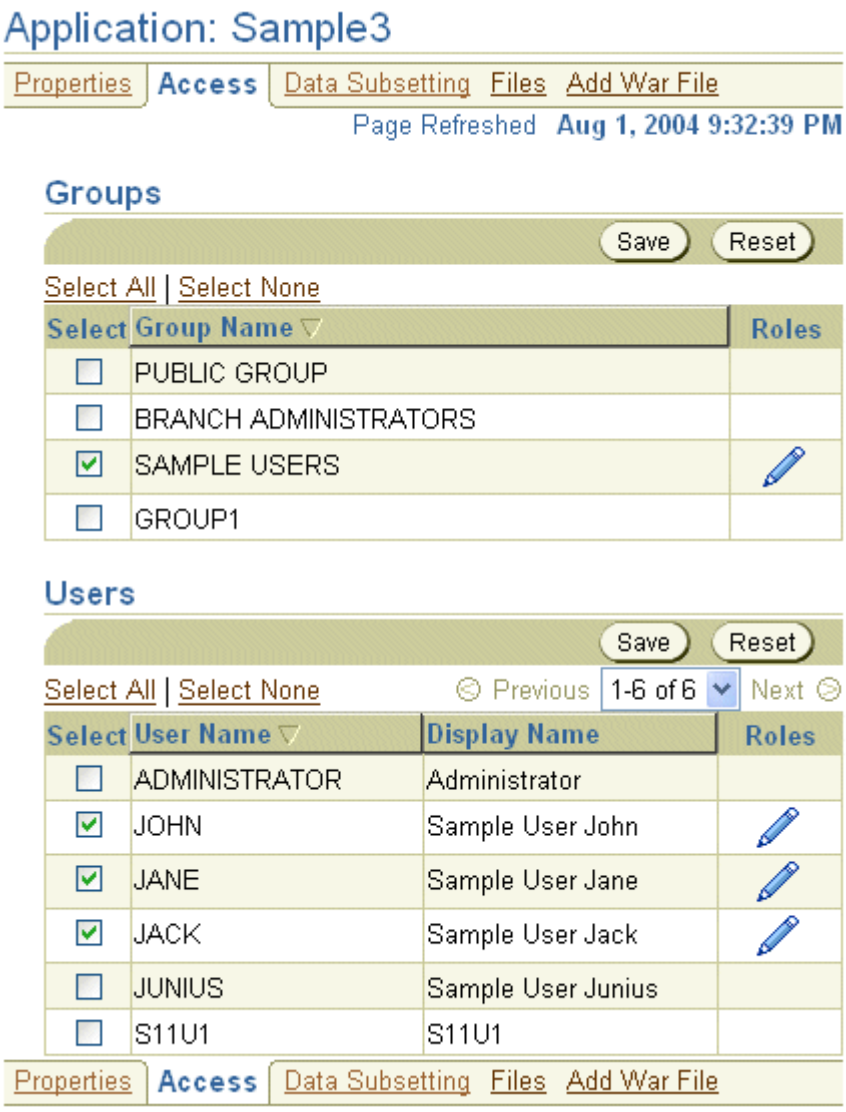
The administrator can grant access to applications for specific users within the Mobile Manager, as follows:

1. Navigate to the Applications page. Click the specific application that you wish to modify. The Properties page appears.
2. Click **Access**. The Access page displays a list of users and groups for this application.
3. Select the checkbox next to each user or group that you wish to give access to for this particular application.
4. Click **Save**.

As Figure 3–3 displays, the Access page displays a list of available users and groups for the Sample3 application. Select the users or groups that you want Sample3 to have access to and click **Save**. In this example, the administrator granted access for the Sample3 application to the SampleUsers group and to the users: John, Jane, and Jack.

Note: Once you provide access to a group, all users in that group have access to this application.

Figure 3–3 Granting Application Access



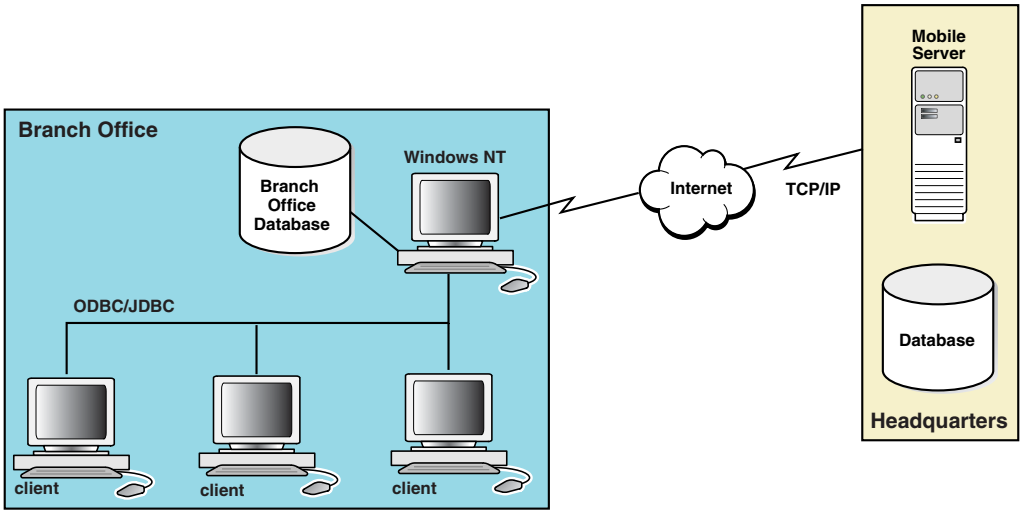
3.6.2 Revoke Application Access to Users and Groups

To revoke application access to any user or group, clear the check box displayed against a group or user name and click **Save**.

3.7 Selecting Application Files for Public Use

To use the Branch Office, you install the Mobile client software on the Branch Office itself. Then, as shown in Figure 3–4, Branch Office maintains its own clients—which are not Mobile clients— by downloading the application onto its clients, which, in turn, communicate directly with the Branch Office. For an overview on what Branch Office is and how to use it, see Chapter 8, "Manage Your Branch Office". For details on setting up a Branch Office and its clients (requiring you to make certain application files public), see Section 8.2, "Branch Office Installation and Configuration".

Figure 3–4 A Branch Office Environment



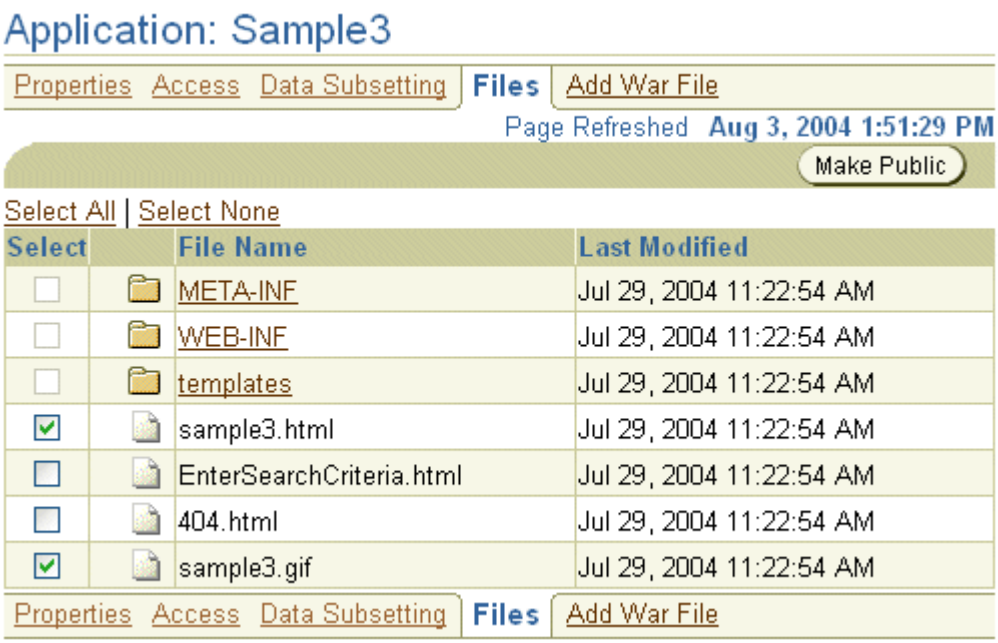
How do you enable the Branch Office to create its clients? The application that is to be executed on the Branch Office clients is published to the Mobile Server. The application, often an executable, that is to be installed on the client is exposed as a public file. The Branch Office downloads the public application executable and is able to install this application on its clients.

Do the following to make the application installation file public.

1. Navigate to the Applications page and click the application link. The Applications home page appears.

   Click **Files**. As Figure 3–5 displays, the Files page lists application files that are assigned for public use.

Figure 3–5 Files Page



2. Select the check box against the application file that you want made public and click **Make Public**

Synchronize the Branch Office, which was previously set up with the appropriate Mobile client software. This brings down the application, the data for this application, and the public application installation file.

On the Branch Office, copy and execute the application public installation file on each Branch Office client. This installs the application on each Branch Office client.

Users can download public files from the Branch Office through the following URL.

```
http://<client>/public/download
```

For full instructions and details, see Section 8.2, "Branch Office Installation and Configuration".

3.8 Adding Web Application Archive (WAR) Files

Using the Mobile Manager, you can add WAR files to your Mobile applications. In accordance with J2EE specifications, you can add Web components to a J2EE application in a package called a Web Application Archive (WAR). It contains all files that make up a Web application including other resources.

To add a WAR file, navigate to the Applications page and click the required application link. The Application Properties page appears. Click the **Add WAR File** link. As Figure 3–6 displays, the Add WAR File page appears.

Figure 3–6 Add WAR File Page

To upload the WAR file, click **Browse** and locate the WAR file. Click **Upload**. You are returned to the Add WAR File page.

3.9 Modifying Registry Entries

If you have used registry entries in the past, you can enable them in the Mobile Manager by adding the `REGISTRY_TAB` parameter to the `webtogo.ora` file. Once you have located the `webtogo.ora` file, enter your registry value in the `REGISTRY_TAB` parameter.

4

Managing Users and Groups

This chapter describes how to manage users and groups using the Mobile Manager. The following topics are covered in this chapter:

4.1 Managing Users and Groups

The following sections discuss how to manage users. Topics include:

4.1.1 What Are Mobile Server Users?

The Administrator or User of Mobile Server user types are described in the following sections:

> **Note:** Do not confuse Mobile Server users with database users. Each Mobile Server user is authenticated by the Mobile Server for access to applications and appropriate publications. The Mobile Server users are not used to access data on the database.

4.1.1.1 Mobile Server User Privilege: Administrator

Any user created with the user privilege of administrator can perform any of the following functions:

- The administrator user can be a general user when logging in to a Mobile application on a device, which is the same as described in Section 4.1.1.2, "Mobile Server User Privilege: User".
- The administrator can publish applications either through the Packaging Wizard or through the Mobile Manager.
- The administrator has authorization to use the Mobile Manager.

Once an administrator user is created, it must be associated with the Mobile Manager in the same manner that an ordinary Mobile Server user is associated with any application. See Section 4.1.1.3, "Associating Mobile Server Users With Published Applications" for more information on this process.

4.1.1.2 Mobile Server User Privilege: User

The Mobile Server user with privilege of user is created only for accessing and synchronizing published applications and its data. The user has a specific username/password for synchronizing the application from a device.

Note: The username or password is limited to a maximum of 28 characters long.

Thus, this Mobile Server user enables access to a particular Mobile application and its publication items. That is, in order for the Windows CE or other devices to be able to synchronize and retrieve a snapshot of data from the database, the Mobile Server validates that the username/password that is entered is valid for the application. If it is, then Mobile Server enables the device to retrieve the snapshot that is indicated by the publication items packaged with the application.

After creating the user, the administrator associates the user with the published applications from which this user will receive data. In addition, if any of the publication items require a parameter to be set, the administrator also sets this parameter for each user. See Section 4.1.1.3, "Associating Mobile Server Users With Published Applications" for more information.

4.1.1.3 Associating Mobile Server Users With Published Applications

Any user that wants to use an application must be associated with that application by an administrator user in the Mobile Manager. In order to associate Mobile Server users with applications, a Mobile Server administrator performs the following:

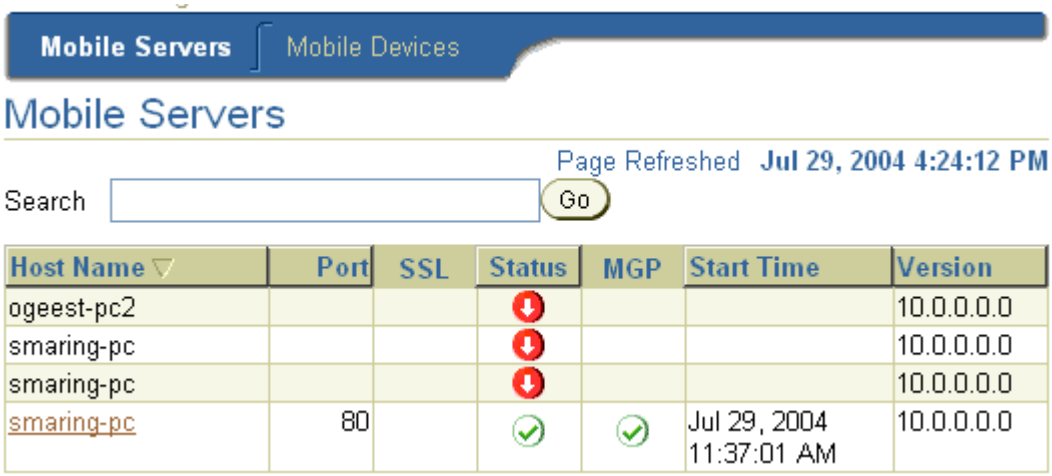
1. Package and publish an application with appropriate publication(s).
2. Create one or more users or groups that will use the application to retrieve data from the database down to a device. See Section 4.1.3, "Adding New Users" for more information.
3. Associate the users or groups with the application. See Section 4.2.1, "Grant or Revoke Application Access to Users" for more information.
4. Optionally, if the application has parameters, also known as data subsetting, that are set for each user or group, define these parameters for each user or group. See Section 4.3, "Managing Application Parameter Input (Data Subsetting)" for more information.

4.1.2 Displaying Users

You can see what users and groups have been created with all information relevant to users—such as user names and so on.

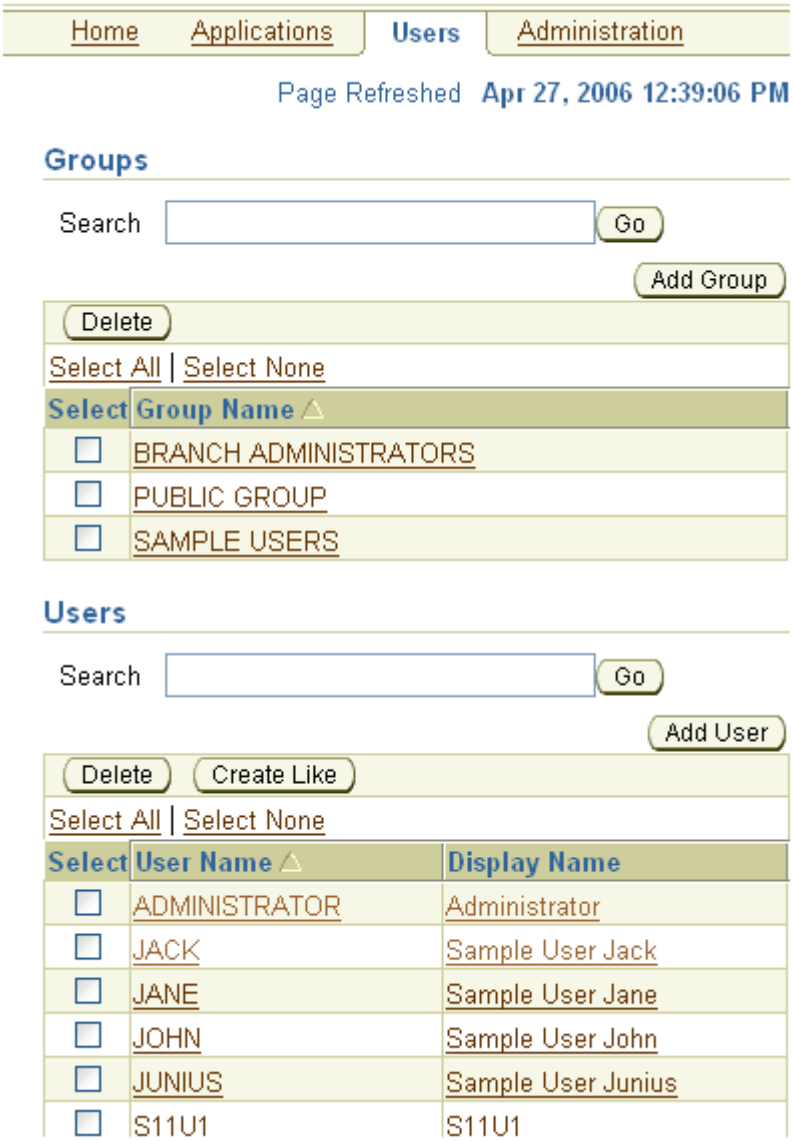
To display individual users, logon to the Mobile Manager and click the **Mobile Manager** link in the Workspace. As displayed in Figure 4–1, the Mobile Servers Farm page is displayed.

Figure 4–1 Mobile Server Farms Page



Click your Mobile Server name link. Your Mobile Server home page appears. Click the **Users** link. As Figure 4–2 displays, the Users page lists existing groups and individual users.

Figure 4–2 Users Page



4.1.2.1 Enabling OID Users

By default, the users defined for access within Mobile Server are contained within the Mobile repository. However, you can specify to use OID as the repository for all users. In this case, you can migrate any existing users from the Mobile Server repository into OID. For details on using OID, see Section 4.1.7, "Managing OID Users in the Mobile Server"; for details on how to migrate users to OID, see Section 6.2.7, "Migrate Your Users From the Mobile Server Repostory to the Oracle Internet Directory" in the *Oracle Database Lite Getting Started Guide*.

Mobile Server is aware of which users were migrated into OID and marks them as "enabled" for use within Oracle Database Lite. By default, all users created within OID are not "enabled" for use within Oracle Database Lite. All OID users are displayed, but are not enabled for Mobile Server. You can enable these users within OID by checking the Enabled box next to the name on the Users screen. This box is only displayed in the case where OID is used as the repository for the users.

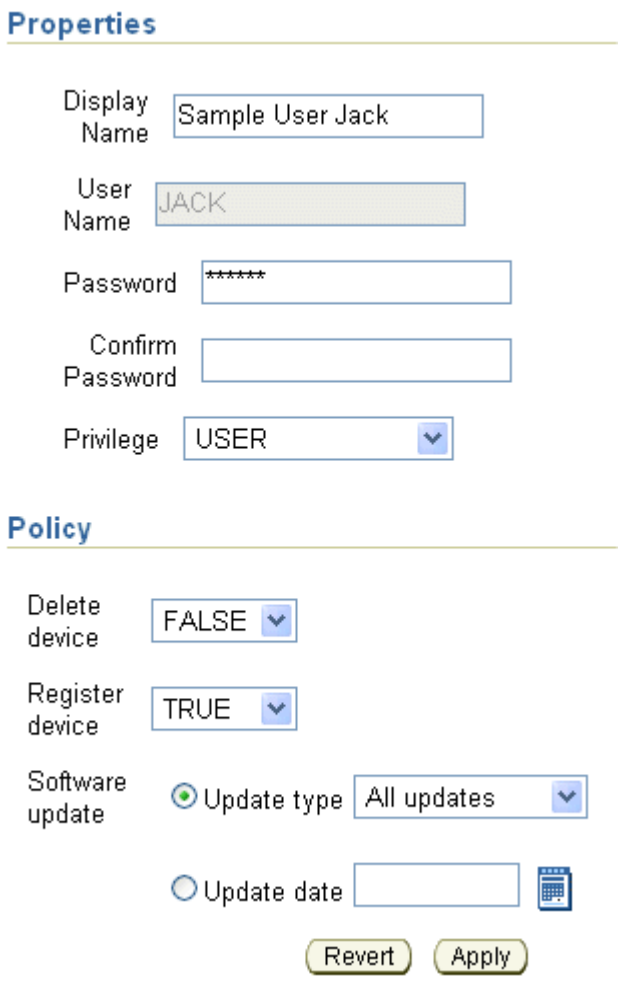
4.1.2.2 Searching Group Names or User Names

To search for a group name or individual user name, enter the group name or user name in the **Search** field and click **Go**. The Users page displays the search result under the Group Name or User Name column.

4.1.3 Adding New Users

To add a new user, navigate to the Users page and click **Add User**. As Figure 4–3 displays, the Add User page appears and lists the requisite criteria to register user properties.

Figure 4–3 Add User Page



To register user properties for new users, enter the following:

- Section 4.1.3.1, "Define Username and Password"
- Section 4.1.3.2, "Assign Priviledge"
- Section 4.1.3.3, "Specify Device Policy"

4.1.3.1 Define Username and Password

To add a new user, enter data as described in the following table.

Table 4–1 Add User Page Description

Field	Description
Display Name	Name used to display as Mobile Server user name.
User Name	Name used to logon to the Mobile Server.
Password	Optional. Password used to logon to the Mobile Server.
Password Confirm	Optional. To confirm the above mentioned password, re-enter your password.
Privilege	Lists available privileges for the Mobile Server user. ■ The **Administrator** option provides privileges to modify Mobile Server resources. ■ The **User** option provides access for registered users to the Mobile Server. For a description of each privilege type, see Section 4.1.1, "What Are Mobile Server Users?" and Section 4.1.3.2, "Assign Priviledge".

Note: User names and passwords can only contain single-byte characters and cannot contain characters such as ', ", @ ,% or blank spaces.

4.1.3.2 Assign Priviledge

Users can be assigned the following privileges.

- Administrator: Manages the Mobile Server and its components, publishes and manages applications, and provides application access to groups and users. See Section 4.1.3.2.1, "Administrator" for more information.
- User: Accesses published applications. See Section 4.1.3.2.2, "User" for more information.

4.1.3.2.1 Administrator Once an administrator user is created, it must be associated with the Mobile Manager in the same manner that an ordinary Mobile Server user is associated with any application. The Mobile Manager is similar to any other mobile application. It provides the following privileges to the administrator.

- To logon to an application on a device, the administrator can use administrator as the user name and password.
- The administrator can publish applications either through the Packaging Wizard or through the Mobile Manager.
- The administrator has authorization to use the Mobile Manager.

4.1.3.2.2 User The Mobile Server user is assigned user privileges and is created for being associated with published applications. The user is provided a user name and password for logging in to an Oracle Lite client and accessing applications from a device. When a user synchronizes with the Mobile Server, the Mobile Server validates the user name and password that is provided by a user and downloads the corresponding applications and snapshots to the client.

After creating a user, the administrator associates the user with a published application. The user can then access such applications and receive data. If any of the publication items require a data subsetting parameter to be set, the administrator sets this parameter for each user.

4.1.3.3 Specify Device Policy

Specify the device policy as follows:

- **Register Device**: To indicate device registration for the group, select True.
- **Software Update**: To indicate the device software update type, select the appropriate option. For example, to update the user's devices with major updates, select this option. To indicate the update date, select the date picker and choose the software update date.

To add the new user and record the device policy, click **OK**.

4.1.4 Duplicating Existing Users

You can duplicate the privilege and device policy of an existing user in creating a new user. On the main User page, as shown in Figure 4–2, select the user that you want to duplicate and then click Create Like. This brings you to a screen where you can enter the following:

Table 4–2 Add User Page Description

Field	Description
Display Name	Name used to display as Mobile Server user name.
User Name	Name used to logon to the Mobile Server.
Password	Optional. Password used to logon to the Mobile Server.
Password Confirm	Optional. To confirm the above mentioned password, re-enter your password.

For more information on privileges and device policy, see Section 4.1.3, "Adding New Users".

4.1.5 Deleting Groups or Individual Users

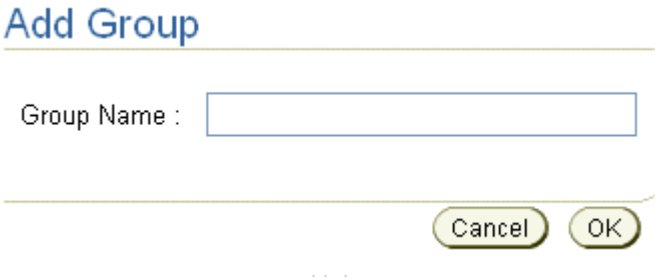
As an administrator, you can delete groups or individual users from the system. To permanently delete groups or individual users from the system, select the **Delete** check box against the group name or individual user name that you want to delete, and click **Delete**. The Mobile Manager seeks your confirmation to delete the chosen group or user name. Click **Yes**. You will be returned to the Users page.

4.1.6 Adding New Groups

If you have several users that require access to the same application, you can bypass adding access rights for each user by including these users in a group. Once all of the users are included in a group, then assign access to the intended application to the group; at this point, all users in the group have access to the application.

As an administrator, you can add a new group that accesses the Mobile Server. To add a new group, navigate to the Users page and click **Add Group**. As Figure 4–4 displays, the Add Group page appears and lists the requisite criteria to register user group properties.

Figure 4–4 Add Group Page



Enter the new group name in the **Group Name** field and click **OK**.

4.1.7 Managing OID Users in the Mobile Server

If you want, you can use the Oracle Internet Directory (OID) for storing and retrieving user information instead of the Mobile Server Repository. To facilitate using OID, you must first migrate all user information from the repository into OID. Once migrated, you can use OID instead of the repository.

OID is part of the OracleAS application server.

If you decide to use OID users (from OracleAS), then after you install the application server and Oracle Mobile Lite, perform the following:

1. If you currently have installed the Mobile Server and have existing users in the Mobile Server, then you must migrate any existing Mobile users to OID (See Section 3.2.7, "Migrate Your Users From the Mobile Server Repository to the Oracle Internet Directory" in the *Oracle Database Lite Getting Started Guide*).
2. Set the `SSO_ENABLED` parameter in the `webtogo.ora` file to `YES`. In the Mobile Manager, migrate Administration tab and select **Edit Config file**. This is the `webtogo.ora` file.
3. Restart the application server. When you modify the `SSO_ENABLED` parameter, the Mobile Server modifies the application server configuration.
4. Enable OID users for the Mobile Server. See Section 4.1.2.1, "Enabling OID Users".

   Note: When you navigate to the Users page in the Mobile Manager, all OID users are displayed. Add any new users through OID. On this page, you can only enable OID users for use within the Mobile Server or change the password.

   To enable OID users for the Mobile Server, select the user and click **Enable**.
5. Assign the appropriate application to these users. As with any Mobile Server user, you must grant access to the appropriate applications. See Section 4.2.1, "Grant or Revoke Application Access to Users" for more information.

4.2 Managing Access Privileges for Users and Groups

The Mobile Server Administrator grant access privileges to Mobile applications by designating the users that can access these applications. The following sections describe the access feature of the Mobile Server:

- Section 4.2.1, "Grant or Revoke Application Access to Users"

- Section 4.2.2, "Include or Exclude Users from Group Based Access"
- Section 4.2.3, "Grant or Revoke Application Access to Groups"

4.2.1 Grant or Revoke Application Access to Users

The following sections describe how an administrator can grant or revoke application access to users and groups:

- Section 4.2.1.1, "Grant Application Access to Users"
- Section 4.2.1.2, "Revoke Application Access to Users"

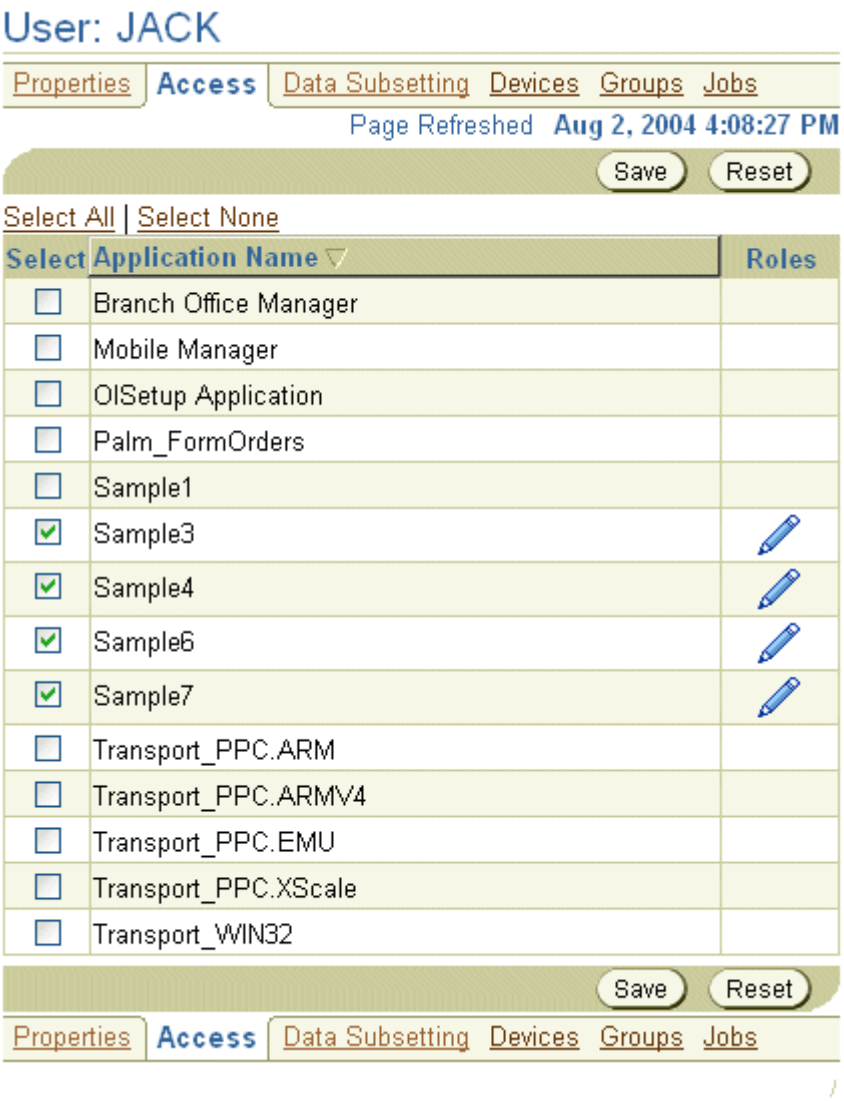
4.2.1.1 Grant Application Access to Users

The administrator can grant access to applications for specific users within the Mobile Manager, as follows:

1. Navigate to the Users page. Click the specific user name to which you wish to give access. This user's Properties page appears.
2. Click **Access**. The Access page displays a list of published applications.
3. Select the checkbox next to each application that you wish to give access to for this particular user.
4. Click **Save**.

As Figure 4–5 displays, the Access page displays a list of available applications for the user Jack. Select the applications that you want Jack to have access to and click **Save**. In this example, Jack is given access to Sample1, Sample3, Sample4, Sample6, and Sample7 applications.

Figure 4–5 Granting Application Access



4.2.1.2 Revoke Application Access to Users

To revoke application access to any user, clear the check box displayed against an application name and click **Save**.

> **Note:** Granting application access to an entire group gives each user in the group, access to the application. For directions on how to include or exclude any user from a group, see Section 4.2.2, "Include or Exclude Users from Group Based Access".

4.2.2 Include or Exclude Users from Group Based Access

The following sections describe how the Administrator can include or exclude users from group based access:

- Include Users in a Group
- Exclude Users from a Group

Using the Mobile Manager, you can modify group based access privileges to include or exclude users requiring access to Mobile applications. To modify group based access privileges, click the **Users** link. The Users page lists existing groups and individual users.

Include Users in a Group

To include users into a group, do the following:

1. Navigate to the Users page. Click the username of the user you wish to include in a group. The user Properties page appears.
2. Click **Groups**.
3. Select the group name that you want to include the user into.
4. Click **Save**.

> **Note:** Existing users with privileges for group based access only can be excluded from group based access.

Now the user takes on the access for all applications to which the group has access. In order for the group to be given access to additional applications, follow the instructions in Section 4.2.1, "Grant or Revoke Application Access to Users". However, instead of selecting a particular user, select the group instead.

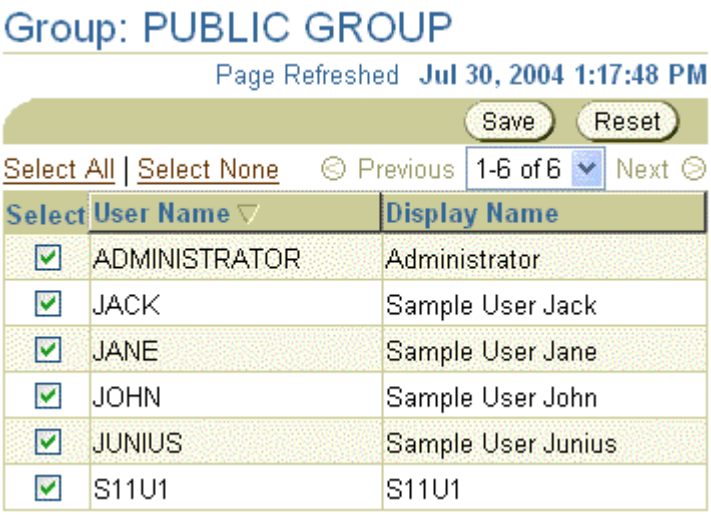
Exclude Users from a Group

To remove a user from any group, do the following:

1. Navigate to the Users page. Click on the username of the user you wish to exclude from a group. The user Properties page appears.
2. Click **Groups**.
3. Clear the group name that you want to exclude the user from.
4. Click **Save**.

Figure 4–6 displays the Clear Group page for the Public Group. If you wanted to clear Jack from this group, you would uncheck the checkbox next to Jack's name and click Save.

Figure 4–6 Clear Group Page



4.2.3 Grant or Revoke Application Access to Groups

Once you have the users that you want in a group, you must indicate what applications that the group has access to. In order to assign application access to groups, you have to add the access rights off the application page. See Section 3.6.1, "Grant Application Access to Users and Groups" for directions.

4.3 Managing Application Parameter Input (Data Subsetting)

If the application that this user accesses requires one or more parameters to determine what data is retrieved from the database, you set these parameters, also known as data subsetting, within the user configuration in Mobile Manager.

Note: You can only set the parameter values once a user has been granted access to the application. See Section 4.2, "Managing Access Privileges for Users and Groups" for instructions.

For example, if you have an application that retrieves the customer base for each sales manager, the application needs to know the sales manager's identification number to retrieve the data specific to each manager. The identification number, in this example, is the application parameter required that is associated with this user. Thus, if you set up each sales manager as a unique user and set their identification number in the data subsetting screen, then the application is given that unique information and can replace it appropriately in the application.

1. Navigate to the Users page. Click the specific user name to which you wish to give access. This user's Properties page appears.
2. Click **Data Subsetting**. The Data Subsetting page enables the administrator to add parameter input for this user. This displays all of the applications that the user is associated with.
3. Select the application for which you want to add the parameter value.
4. Enter the parameter values for the application.
5. Click **Save**.

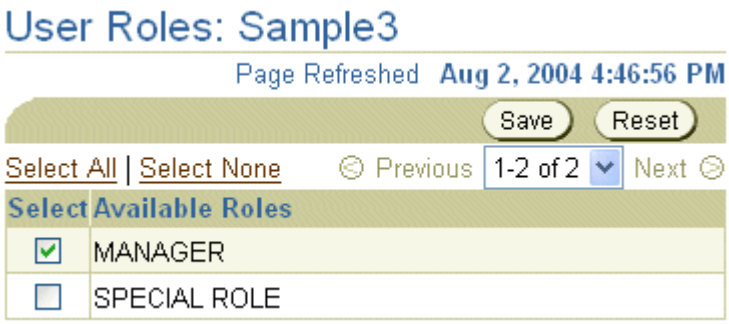
4.4 Assigning Application Roles to Users

When the developers design any OC4J or Web-to-Go application, they can include functionality that is enabled based on the role that the user is assigned. For example, if you have a manager and employee role in an application, the user who is assigned the manager role may have other options available to view on the application GUI. These options would not show up for those users who are assigned the employee role. See Section 7.2.2, "Application Roles" in the *Oracle Database Lite Developer's Guide* for information on how to programmatically create and grant these roles.

Once the application is deployed, all roles are displayed and can be assigned to any user in the Mobile Manager. You can assign roles either through the Mobile Manager or through the `wsh` script. This section describes how to assign users to certain roles for an OC4J or Web-to-Go application.

Figure 4–5 displays the User page for Jack. Notice that there is a column for Roles. If you click the pencil icon in this column, you can see the roles that have been created in the application. For example, if we click on the pencil icon for the Sample3 application, as shown in Figure 4–7, we see that two roles have been created in this application: Manager and Special Role. Select the checkbox next to any of the roles to which you want Jack to be added. In this case, the Manager role is checked, so Jack will be added to the Manager role.

Figure 4–7 Add Jack to the Sample3 Application Manager Role



4.5 Creating an Administrator

As referenced in the previous sections, to create any user, including administrators, you must do the following:

1. Create one or more users or groups that will use the application to retrieve data from the database down to a device. See Section 4.1.3, "Adding New Users" for more information.
2. Associate the users or groups with the application. See Section 4.2.1, "Grant or Revoke Application Access to Users" for more information.
3. Optionally, if the application has a parameter, also known as data subsetting, that is set for each user or group, define the parameters for each user or group. See Section 4.3, "Managing Application Parameter Input (Data Subsetting)" for more information.

Thus, to create an administrator, you would do the following:

1. Create a user with the name of the administrator that you want, with the privilege of administrator.

2. Navigate to the Access tab for this new administrator and check the checkbox next to Mobile Manager.

You now have a new administrator user. You can log into your Mobile Manager with this user's name and password.

4.6 Manually Adding Devices for a User

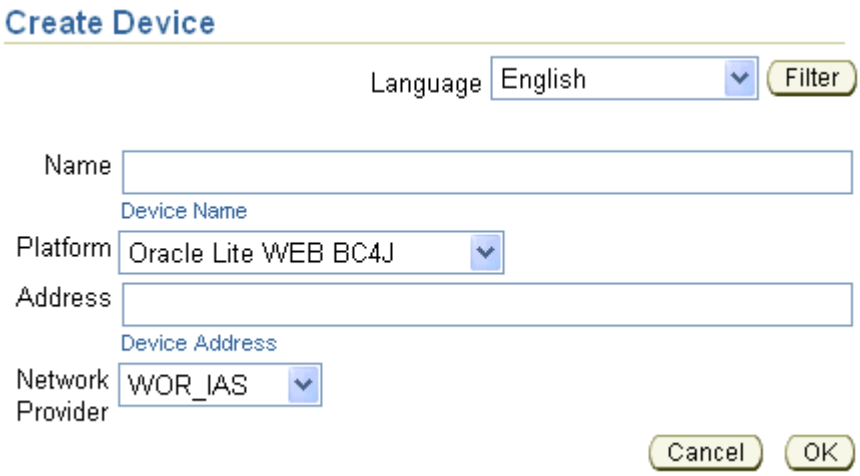
Normally, when you download and install a client, the device is registered automatically for the user. There are two instances where you may need to manually add the device:

- As an administrator, you could hand a device that is fully loaded with the Mobile client software, but is not assigned to any user or application. After handing the device to your user, you can add their user information, application access, and device that they are using manually.
- When you hand someone the Mobile client software on an installation CD, then the installation does not register the device manually—since it is not connected to Mobile Server. Thus, for each user that you provide the Mobile client software from an install CD, you will have to add the device to this user.

To add a device for an individual user, navigate to the specific user's page and perform the following:

1. On the Users page, select the user for which you want to add a device.
2. Click **Devices**. All currently registered devices for this user appear.
3. Click **Add**. The Create Device screen (as shown in Figure 4–8) appears.

Figure 4–8 Manually Add Device to User



4. Enter the device information, as described in Figure 4–3, and click **OK** to add the device for this user:

Table 4–3 Device Information

Device Field	Description
Language	Select the language that the platform will use. The default is English.
Name	Configure a user-defined name for the device.

Table 4–3 (Cont.) Device Information

Device Field	Description
Platform	Select the platform for this device.
Address	The device address indicates the unique network identifier of a device. The device address must have a corresponding Network Provider associated with it. To transmit data to a device, the DMS uses the Network Provider associated with the address object. For example, RAPI, HTTP, WOR, SMTP. To enable a communication link between the DMS and the DMC, the Administrator must create a proper device address for all devices. In the Address field, enter the device address.
Network Provider	To specify the network provider, click the Network Provider box and choose the required network provider from the list displayed.

Once added, the user can now synchronize the device to retrieve their applications and related snapshots.

4.7 Set Update Policy for Software Updates for the User

You can control whether a new version of an application software is downloaded on each client. Modify the update policy attribute of the user with the Software Update pulldown to the appropriate update that you want, as follows:

- All updates—Include major and minor updates.
- Major—The devices attached to this user receives only major software updates. This is the default.
- Minor—The devices attached to this user receives only minor software updates.
- Disable updates—The devices attached to this user does not receive any software updates.

In addition, you can specify the date that the update occurs.

5

Managing Synchronization

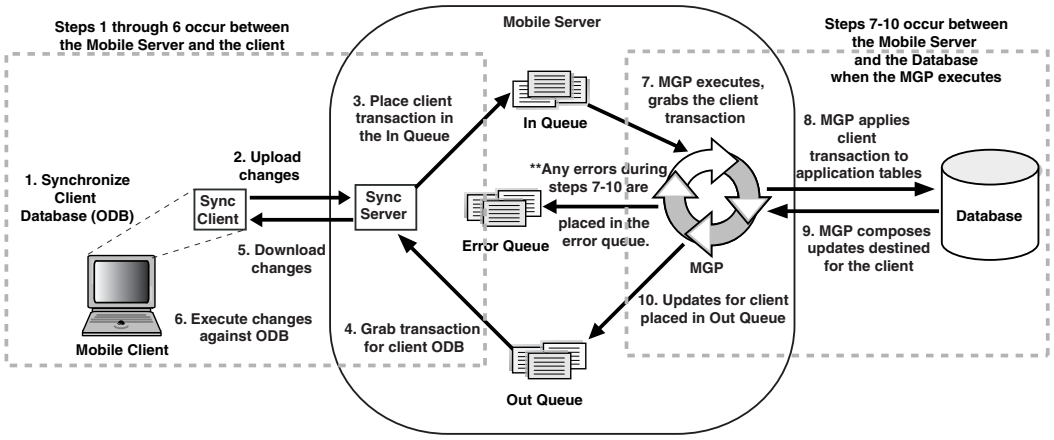
The Mobile Server administrator uses the Data Synchronization Manager to manage synchronization tasks. This chapter includes:

5.1 How Does the Synchronization Process Work?

The Mobile Server uses synchronization to replicate data between the Mobile clients with their client Oracle Lite databases (including those for OC4J, Web-to-Go, Win32, Windows CE, Symbian and Linux platforms) and the application tables, which are stored on a back-end Oracle database.

When most people think of synchronizing data, they think of their Palm Pilot. When you hit the synchronization button for the Palm Pilot, any changes are added to the database of information on the Windows machine immediately. This is not the case for Oracle Database Lite, which is used for multiple clients. In order to accomodate multiple users, the application tables on the back-end database cannot be locked by a single user. Thus, the synchronization process involves using queues to manage the information between the Mobile clients and the application tables in the database (demonstrated by Figure 5–1), as follows:

Figure 5–1 Data Synchronization Architecture



1. Synchronization is initiated on the Mobile client either by the user or from automatic synchronization. Note that the Mobile client may be a Windows platform client or a PDA.
2. Mobile client software gathers all of the client changes into a transaction and the Sync Client uploads the transaction to the Sync Server on the Mobile Server.
3. Sync Server places the transaction into the In-Queue.

> **Note:** When packaging your application, you can specify if the transaction is to be applied at the same time as the synchronization. If you set this option, then the transaction is immediately applied to the application tables. However, note that this may not be scaleable and you should only do this if the application of the transaction immediately is important and you have enough resources to handle the load.

4. Sync Server gathers all transactions destined for the Mobile client from the Out-Queue.
5. Sync client downloads all changes for client Oracle Lite database.
6. Mobile client applies all changes for client Oracle Lite database. If this is the first synchronization, the Oracle Lite database is created.

> **Note:** For information on what Oracle Lite database (ODB) files are installed on the client, see Section 2.2, "Synchronize or Execute Applications on the Mobile Client".

7. All transactions uploaded by all Mobile clients are gathered by the MGP out of the In-Queue.
8. The MGP executes the apply phase by applying all transactions for the Mobile clients to their respective application tables to the back-end Oracle database. The MGP commits after processing each publication.

Note: The behavior of the apply/compose phase can be modified. See Section 5.1.1, "Defining Behavior of Apply/Compose Phase for Synchronization" for more information.

9. MGP executes the compose phase by gathering the client data into outgoing transactions for Mobile clients.
10. MGP places the composed data for Mobile clients into the Out-Queue, waiting for the next client synchronization for the Sync Server to gather the updates to the client.

Overall, synchronization involves two parties: the Mobile client using the Sync Client/Server to upload and download changes and the MGP process interacting with the queues and the application tables to apply and compose transactions. These are displayed separately in the Data Synchronization section of the Mobile Manager.

On the Mobile Server home page, you can navigate to the Data Synchronization home page by clicking Data Synchronization, which is located under the Components section.

5.1.1 Defining Behavior of Apply/Compose Phase for Synchronization

By default, before the MGP processes the Compose, it checks to see if user data has been uploaded into the In Queue. The MGP checks to see if the user uploaded data before it performs the compose for that user, because if the compose completes with unresolved data from the user, then the user data may be compromised. So, the Compose is not performed to ensure that user data is not overwritten. Instead, the Compose phase is terminated and then waits until the next time that the MGP runs the Apply/Compose phase.

However, you can modify this behavior. The compose phase may take a while, depending on the number of users, so you may not want to wait until the next MGP compose phase. In this case, use the `DO_APPLY_BFR_COMPOSE` parameter. Or, maybe you know that the uploaded client data will not be compromised by the compose; in this case, use the `SKIP_INQ_CHK_BFR_COMPOSE` parameter.

Table 5–1

Webtogo.ora Parameter	Description
`DO_APPLY_BFR_COMPOSE`	Setting `DO_APPLY_BFR_COMPOSE` to `true` modifies this behavior. If data is in the in queue, MGP will execute a second Apply to commit all user data and then will execute the Compose.
`SKIP_INQ_CHK_BFR_COMPOSE`	Setting `SKIP_INQ_CHK_BFR_COMPOSE` to `true` modifies this behavior. Even if data is in the in queue, MGP executes the Compose. The data that was uploaded to the In Queue must be data that will not be compromised by downloading data from the server to the client.

Note: Setting these parameters can also avoid the MGP Compose postponed error. For more information, see Section 2.1.6 "MGP Compose Postponed Due to Transaction in the In-Queue" in the *Oracle Database Lite Troubleshooting and Tuning Guide*.

5.2 User Scenarios for Synchronization

The following scenarios demonstrate how a client user may want to synchronize the data:

- You can enable Automatic Synchronization between the client and the server, which is specified on the publication item level. With automatic synchronization, you can specify under which conditions synchronization is automatically started to save any changes on the client back to the server. This way, the client data is synchronized on a regular basis in the background, automatically, without user intervention.

  For more information on automatic synchronization, see Section 5.4, "Using Automatic Synchronization".

- You can specify that the client or the client application manually synchronizes. The user can synchronize through a GUI; an application can initiate synchronization programmatically through the APIs. This manually initiates synchronization for uploading/downloading the modifications made on the client and server. This is the default mechanism for synchronization.

  – If the user is going to start the synchronization, use the GUI tools, as described in Section 2.2, "Synchronize or Execute Applications on the Mobile Client".

  – If the application is going to initiate the synchronization, use the synchronization APIs, as described in Chapter 5, "Invoking Synchronization APIs from Applications" in the *Oracle Database Lite Developer's Guide*.

- You can enable a type of synchronization where only the data on the client is uploaded to the server; data is never downloaded from the server. This is an option for read-only clients, where multiple clients are using the same data. If you have a situation where you have a large number of clients that use the same read-only data, use the cached user, which can be replicated on multiple clients. See Section 5.6, "Improving Performance for Multiple Clients that Use the Same Read-Only Data With a Cached User" for more details.

5.3 Managing the Sync Server from the Data Synchronization Home Page

The Sync Server is an HTTP servlet that listens to client synchronization requests. As demonstrated by Figure 5–1, during every synchronization session, the Sync Server uploads client transactions from the client Oracle Lite database and places them within the In-Queues. The Sync Server then downloads any server-side transactions from the Out-Queues to the client Oracle Lite database.

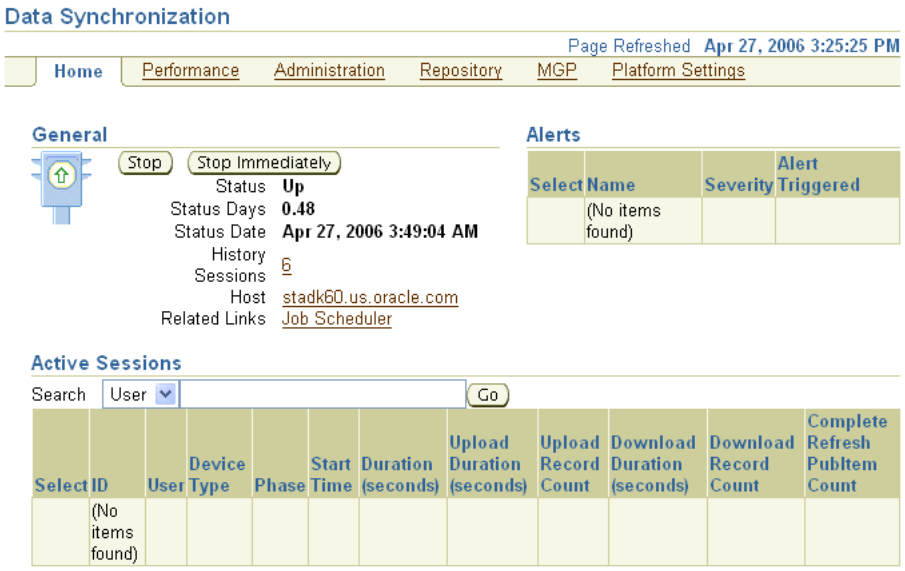
From the Data Synchronization home page, you can manage Sync Server tasks—such as the following:

- Section 5.3.1, "Starting/Stopping the Sync Server"
- Section 5.3.2, "Checking Synchronization Alerts"
- Section 5.3.3, "Managing Sync Sessions"
- Section 5.3.4, "Displaying Operating System (OS) and Java Virtual Machine (JVM) Information"

5.3.1 Starting/Stopping the Sync Server

To start the Sync Server, navigate to the Data Synchronization home page. The Sync Server default status is Up, as displayed in Figure 5–2.

Figure 5–2 Data Synchronization Home Page



To gracefully shut down the Sync Server, click **Stop**. The Sync Server stops after all current sessions have completed synchronization. To immediately stop the Sync Server, click **Stop Immediately**, which kills current sync sessions immediately. Use for emergency situations.

5.3.2 Checking Synchronization Alerts

On the right-hand side of the Data Synchronization Home page, you can see all of the alerts. Both the Sync Server and MGP register alerts if a problem occurs within any part of the synchronization phases. There are two types of alerts, as follows:

- Critical alerts—For the Sync Server, clients cannot synchronize if the Sync Server encounters an exception (also known as a critical alert); thus, the errors must be resolved by the administrator. Once resolved, the administrator re-starts the Sync Server.
- Warning alerts—These alerts are registered when an individual synchronization session fails. The administrator checks the Sync session details in the Sync Session History and determines the reasons for the failure. If necessary, the administrator may need to involve a DBA, if the reason is database-related.

Each alert provides the alert name, degree of severity, time when the alert was triggered, and time when the alert was last checked by a DBA.

Table 5–2 lists sample alerts. Note that the type designates whether the alert originates from the Sync Server or the MGP.

Table 5–2 Alert Types

Name	Type	Severity
`Sync Server Exception`	Sync Server	`CRITICAL`
`User Sync Failure(s)`	Sync Server	`WARNING`
`MGP Job Exception`	MGP	`CRITICAL`
`MGP User Apply/Compose Failure(s)`	MGP	`WARNING`

5.3.3 Managing Sync Sessions

For all users, the sessions that are currently in the process of synchronization are displayed in the Active Sessions table at the bottom of the Data Synchronization Home Page. Synchronization involves uploading or downloading updates between the the Sync Client and Sync Server.

You can terminate any active session on the Data Synchronization Home page by performing the following steps:

1. Select the active session that you wish to terminate and click **Kill**.
2. Click **Yes**.
3. Click **OK** for the confirmation message.

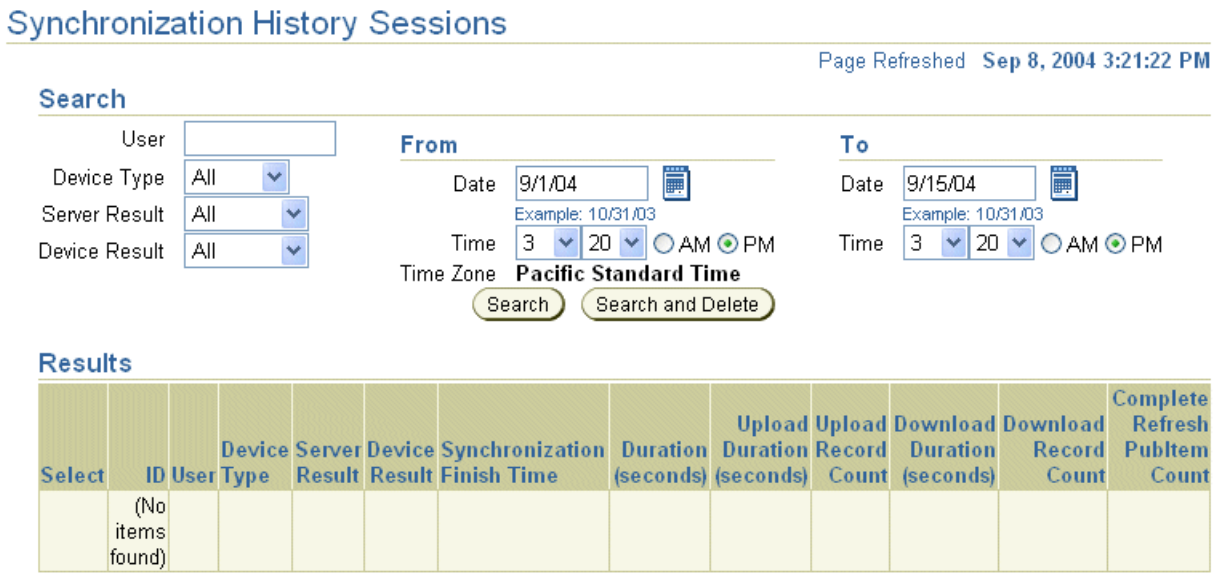
The Active Sessions table on the Data Synchronization home page also displays session details. Select the active session that you wish to view and click **Details** to see the publication items that have been uploaded or downloaded, waiting publication items, records and timing information, and the session trace file.

If you want to view all details about completed synchronization sessions, navigate to the Synchronization History Sessions screen. To navigate to this screen, either click the number hyperlink next to History Sessions on the home page or navigate through the Performance tab. The total number of registered sessions is designated by the number next to History Sessions.

Note: The session history for each user between the Sync Client and Sync Server is saved only if you set the `SYNC_HISTORY` parameter to `YES`, which is the default. You can set the `SYNC_HISTORY` instance parameter to `YES` or `NO` by navigating to `Data Synchronization->Administration->Instance Parameters`.

Figure 5–3 shows the Synchronization History Sessions page.

Figure 5–3 Synchronization History Sessions Page



All session history is not displayed until you search for the appropriate records. If you want all records within a specified date, then the only thing that you need to provide is the From and To date range and click **Search**. Be careful to only click **Search and Delete** if you want these records removed. You can further narrow the search by specifying one or more of the following:

- The name of the user from which all synchronizations originated
- The device platform type to see all synchronizations from just these platforms.
- Only those synchronizations that were successful or failures from the server-side.
- Only those synchronizations that were successful or failures from the device-side.

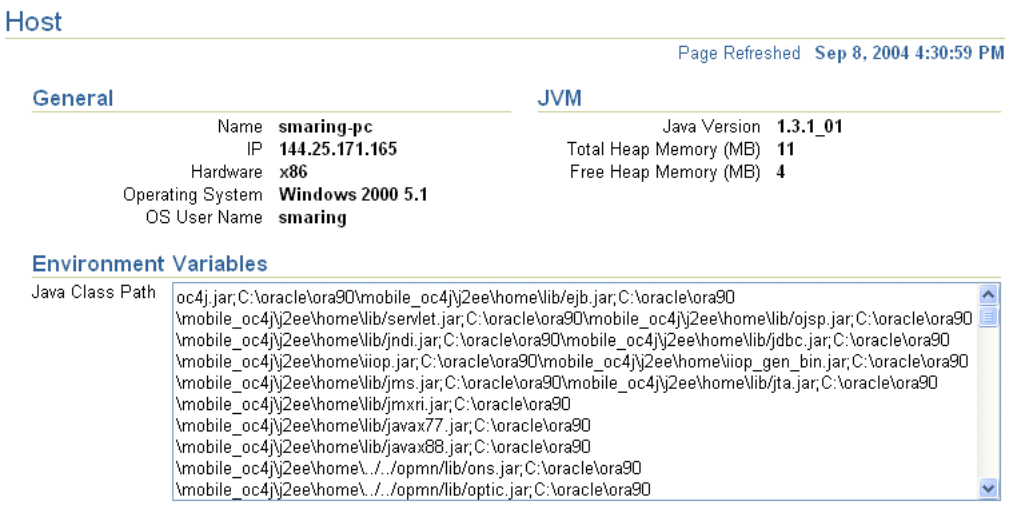
The Session History page displays matched sessions in the **Results** section. Once displayed, you can sort by most of the headers to either sort top to bottom or bottom to top. For example, to sort sync sessions by user, click the **User** header title.

To delete a session, select the session that you want to delete and click **Delete**. To view the details of a session, select the session and click **Details**. The Sync History Session page displays session details, such as publication items that are uploaded or downloaded, records and timing information, and the session trace file. The **View** and **Download** links are automatically enabled for viewing or downloading trace files that are available for the chosen session.

5.3.4 Displaying Operating System (OS) and Java Virtual Machine (JVM) Information

You can see the operating system and JVM versions that are installed on the host where the Mobile Server resides by clicking the **Host** hyperlink that is located below the Start/Stop buttons on the Data Synchronization home page. As displayed in Figure 5–4, the Host page displays host information, such as host name, IP address, OS type, and OS user name. The JVM section displays the Java `CLASSPATH`, Java version, and heap memory size.

Figure 5–4 Host Page



5.4 Using Automatic Synchronization

In the past, a client had to manually request synchronization either through an application program executing an API or by a user manually pushing the Sync button. Now, you are provided the option to configure under what circumstances

synchronization should occur and then Oracle Database Lite performs the synchronization for you automatically. This way, synchronization can happen seamlessly without the user's knowledge.

For example, you may have a user who changes data on their handheld device, but does not sync as often as you would prefer. You may have multiple users who all synchronize at the same time and overload your system. These are just a few examples of how automatic synchronization can make managing your data easier, be more timely, and occur at the moment you need it to be uploaded.

Automatic synchronization is specified on the publication item. The rules for when and how synchronization is automatically started may be specified in the publication, publication item, and/or platform level. With automatic synchronization, the client data is backed up on a regular basis in the background, automatically, without user intervention.

The rules for Automatic Synchronization are defined in three places:

- Enable Automatic Synchronization at the publication item level when creating the publication item.

  For more information on how to enable automatic synchronization at the publication level, see Section 5.4.2, "Start, Stop, or Get Status for Automatic Synchronization" or Chapter 4, "Enabling Automatic Synchronization" in the *Oracle Database Lite Developer's Guide*.

- Define publication-specific rules that apply only to publication items that are enabled for automatic synchronization within this publication. This includes rules that are defined for the data or for specific platforms using this publication.

  For more information on how to specify rules for all enabled publication items, see Chapter 4, "Enabling Automatic Synchronization" in the *Oracle Database Lite Developer's Guide*.

- Define platform-based rules that apply to all publications on a specific platform. This is specified at the platform-level. Thus, see Section 5.4.1, "Specifying Platform Rules for Automatic Synchronization" for more information.

The following provides more information about your automatic synchronization:

- Section 5.4.1, "Specifying Platform Rules for Automatic Synchronization"
- Section 5.4.2, "Start, Stop, or Get Status for Automatic Synchronization"
- Section 5.4.3, "How the Automatic Synchronization Transaction is Retried"

5.4.1 Specifying Platform Rules for Automatic Synchronization

You can specify rules that apply to publications that are enabled for automatic synchronization for a given platform. There are two types of rules: events and conditions. If an event is true, it starts synchronization; however, the synchronization cannot occur unless all conditions are true, as well. This evaluates as follows:

```
when EVENT and if (CONDITIONS) then sync;
```

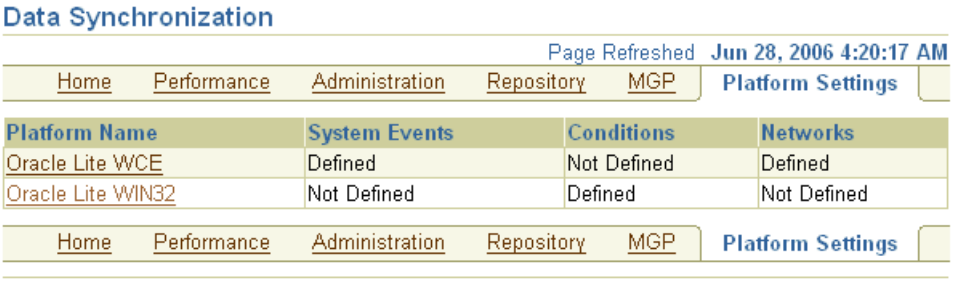
Specify these rules in the Mobile Manager Platform Settings page under Data Synchronization.

> **Note:** These rules only apply to automatic synchronization. If the user manually starts synchronization, it will execute.

To specify platform-based rules for all publications, perform the following:

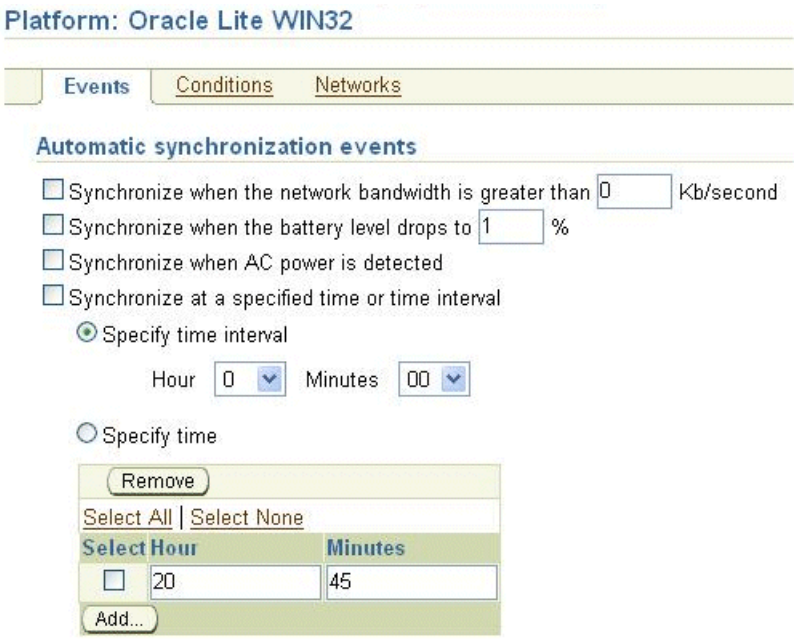
1. On the Data Synchronization page, select **Platform Settings**. Figure 5–5 shows the settings for automatic synchronization on each platform.

Figure 5–5 Platform Settings for Automatic Synchronization



2. Select the platform name to modify the automatic synchronization platform settings. Figure 5–6 shows the screen for the platform-based rules. There are three tabs: System Events, Conditions, and Networks.

Figure 5–6 Platform-Based System Events for Automatic Synchronization



3. Configure the System Events. Figure 5–6 shows the System Events page. Select the checkbox for each event that you want to enable. If the event requires a value, enter the value you desire to be followed. If one event is true, then the automatic synchronization is initiated the first time the event occurs. For example, if the battery runs below the percentage you specified, the automatic synchronization occurs. As the battery continues to deplete, you will not trigger another synchronization.

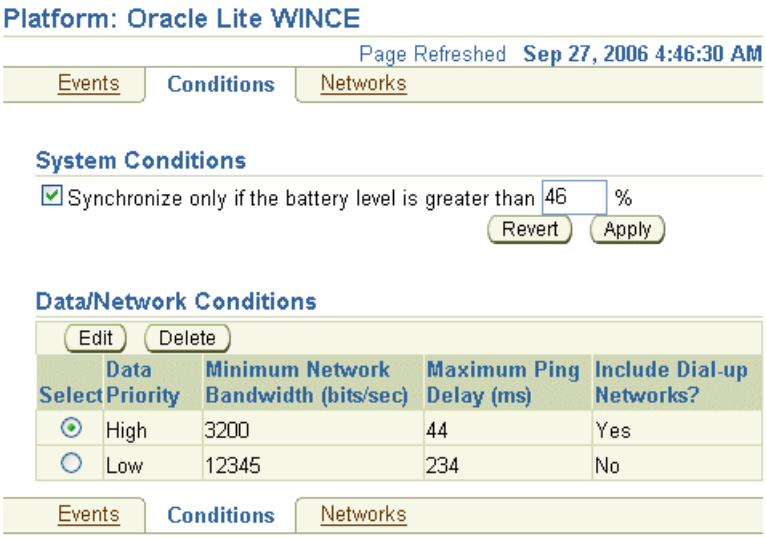
   The following three system events will trigger an automatic synchronization if true.

- Synchronize when the network bandwidth is greater than <number> Kb/second. Where <number> is an integer that indicates the bandwidth in KB/seconds. When the bandwidth becomes available, the synchronization occurs.
- Synchronize when the battery level drops to <number>%, where <number> is a percentage. Often you may wish to synchronize before you lose battery power. Set this to the percentage of battery left, when you want the synchronization to automatically occur.
- Synchronize when the AC power is detected. Select this checkbox if you want the synchronization to occur when the device is plugged in.
- Synchronize at a specific time or time interval. You can configure an automatic synchronization to occur at a specific time each day or as an interval.
  - Select **Specify Time** if you want to automatically synchronize at a specific hour, such as 8:00 AM, everyday.
  - Select **Specify Time Interval** if you want to synchronize at a specific interval. For example, if you want to synchronize every hour, then specify how long to wait in-between synchronization attempts.

4. Configure the Platform Conditions. Select the **Conditions** tab. Figure 5–7 displays the Conditions screen.

   If an Automatic Synchronization is about to start, Oracle Database Lite evaluates the conditions to determine if the synchronization can continue. If the condition is not true, the synchronization cannot proceed. For example, if you enabled that synchronization can only occur if the battery level is greater than 30%, then if an automatic synchronization is about to start, but the battery level is at 20%, this synchronization is canceled.

Figure 5–7 Platform-Based Conditions for Automatic Synchronization



The following conditions that must be true for this platform in order for any automatic synchronization to occur:

- Synchronize only if the battery level is greater than <number>%, where <number> is the percentage of battery level left. Sometimes you may not want synchronization to occur and use up what battery you may have left. Thus,

you can specify a minimum at which point you do not want this feature to occur. This condition must be true in order for an automatic synchronization to occur.

Click **Apply** to save changes; click **Revert** to cancel changes.

- Data/Network Conditions: You could have defined records in your snapshot with a data priority of HIGH (value of 0) or LOW (value of 1).

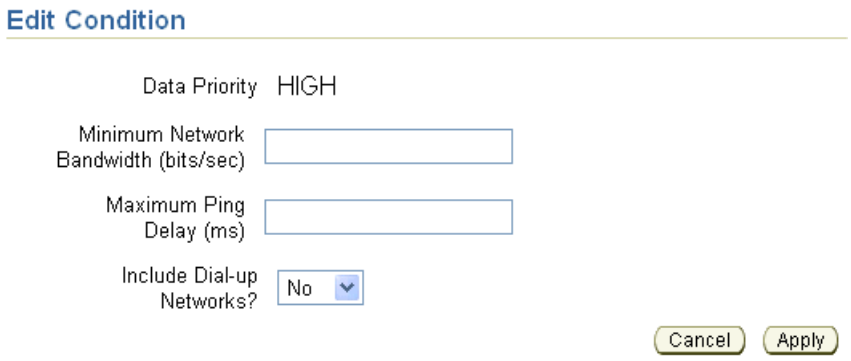
Note: Data priority is a column that is added to the table to indicate priority of the row. You can modify the values in this column to either 0 or 1.

Use this condition to specify 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.

  - Select an existing condition and click **Edit** to modify an existing condition.
  - Select an existing condition and click **Delete** to remove an existing condition.

If you selected a condition and clicked **Edit**, Figure 5–8 displays the fields that you can specify for this condition.

Figure 5–8 Editing the Data Priority Condition

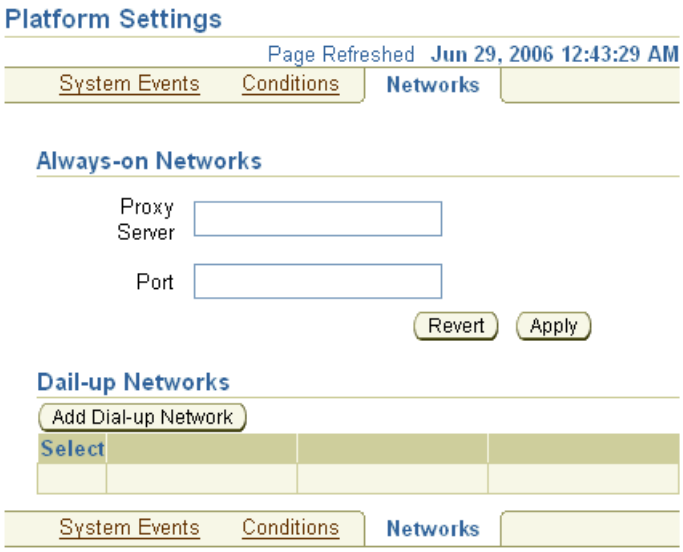


The values you can specify for the data priority condition are as follows:

- Minimum Network Bandwidth (bits/sec): Configure the minimum bandwidth (bits/second) in which the automatic synchronization can occur for records with this data priority.
- Maximum Ping Delay (ms): Configure the maximum ping delay (milliseconds) in which the automatic synchronization can occur for records with this data priority.
- Include Dial-up Networks?: The always-on network is used if available. However, if this network is not available, select **YES** if you want to use any of the dial-up networks for this data priority.

5. Configure the Network settings for the platform rules.

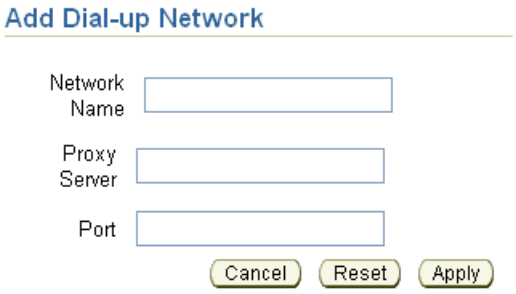
Figure 5–9 Add Network Information for Automatic Synchronization



The Network settings screen provides any proxy server configuration—if your network provider requires that you specify a proxy server to access the internet. You could have two types of networks, as follows:

- Always-on: If this network uses a proxy server, then define the proxy and port number. Click **Apply** when finished.
- Dial-up:
  - Click **Add Dial-up Network** to add a a new entry for dial-up configuration.
  - To edit an existing configuration, select the name of the existing configuration.
  - To delete an existing configuration, select the checkbox next to the desired configuration and click **Delete**.

Figure 5–10 Add Dial-Up Network Information



If you are required to provide proxy configuration for any dial-up network, then configure the following so that Oracle Database Lite can connect to the Mobile Server for the automatic synchronization process. If you do not need to define a proxy for a dial-up network, but you want to include it in the order of execution, you can add an entry with only the network name. You do not need to specify the proxy server and port.

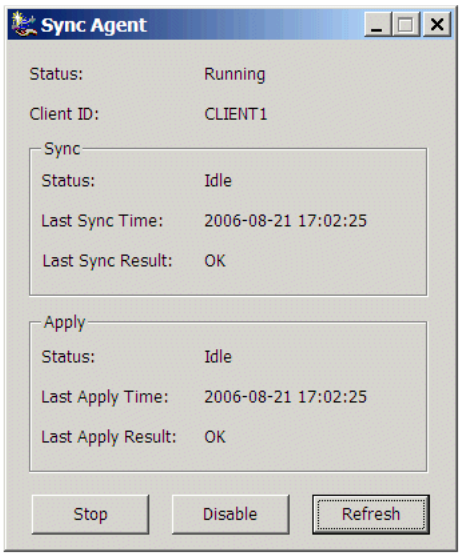
- Network Name—Specify the network name, which is the same as the network name defined on the device.
- Proxy Server—If you have to go through a proxy server, then provide the name of the proxy server for the dial-up network.
- Port—Provide the port number of the proxy server.

5.4.2 Start, Stop, or Get Status for Automatic Synchronization

You can start, stop or retrieve status of automatic synchronization through the Sync Agent UI, which is started either through the Start menu or by running the `syncagent.exe` executable in a command prompt. Figure 5–11 shows the Sync Agent UI.

Note: You can also start and stop automatic synchronization using programmatic APIs. For full details, see Section 3.2.2, "Enable/Disable Automatic Synchronization" in the *Oracle Database Lite Developer's Guide*.

Figure 5–11 Managing Automatic Synchronization Agent



The Synchronization Agent controls the automatic synchronization for the client device. If you want to stop synchronization in order to execute a manual synchronization, click the **Stop** button. This allows any currently executing synchronization to complete fully. If you want to terminate the existing synchronization, click **End**. To restart the automatic synchronization, click **Start**.

Note: If you notice that the automatic synchronization does not start, check the Status to see if the synchronization agent is "Disabled". Enable the synchronization agent with either the `SYNC_AGENT` parameter in `polite.ini` or click **Enable** in the Sync Agent UI.

The start, stop, and end buttons only control the automatic synchronization temporarily. To fully disable automatic synchronization, so that it is not restarted when a device is powered on, click **Disable**. To re-enable automatic synchronization, click

Enable. This can also be accomplished through configuring the `polite.ini`; see Section F.3.2.13, "SYNC_AGENT" for details.

To see the status of any existing or the last automatic synchronization, click **Refresh**.

5.4.3 How the Automatic Synchronization Transaction is Retried

If the automatic synchronization fails because of a network error, the client-side Sync Agent probes the network to retry the transaction, as follows:

- The network is always checked before synchronization is attempted. If the network is not available, the network is checked as follows:
  1. every 15 seconds three times
  2. every 30 seconds three times
  3. every 60 seconds three times
  4. every 20 minutes until the network is available
- If the network is available, then the transaction is retried, as follows:
  1. every 15 seconds three times
  2. every 30 seconds three times
  3. every 60 seconds three times

If the network is still not available, the Sync Agent continues to check the network every 20 minutes. If it detects an available network, the automatic synchronization is started.

5.5 Configuring Data Synchronization For Farm or Single Mobile Server

There are two types of configuration parameters for Data Synchronization:

- Shared—Shared parameters affect all Mobile Server instances in the farm. The administrator can have multiple Mobile Server instances in a single farm that uses the same Mobile repository. To modify these parameters, navigate to the Administration screen and click **Shared Parameters**.
- Instance—Instance parameters only affect a single Mobile Server instance; that is, the Mobile Server that you are currently viewing. These parameters are stored in the `WEBTOGO.ORA` file; thus, once modified, you may need to restart the Mobile Server. Check the Need Restart column to verify if a restart is necessary. To modify these parameters, navigate to the Administration screen and click **Instance Parameters**. See Appendix A, "Configuration Parameters for the WEBTOGO.ORA File" for a description of each of these parameters.

  It is never recommended to modify the `webtogo.ora` file directly; instead, use the Mobile Manager to modify any of the `webtogo.ora` file parameters. In the Mobile Manager, migrate Administration tab and select **Edit Config file**. This is the `webtogo.ora` file.

To view the parameter description and additional information, click **Show**. You can modify any of the values in the **New Value** field and click **Apply**. Some of the Instance parameter values do not take effect until the Mobile Server is restarted.

5.6 Improving Performance for Multiple Clients that Use the Same Read-Only Data With a Cached User

If you have the default method for synchronization, each Mobile client must have a unique synchronization user ID, which is used to coordinate the modifications between the client and the server. However, if you have multiple Mobile clients that use the same data—where the clients only are allowed to read this data—then there is no need to track modifications on the client. Thus, in this scenario, you can choose a more performant solution by configuring a single user for all read-only clients with access to the same publication data.

All Mobile clients share the same user ID, which is subscribed to a publication, where all publication items are read-only. Thus, when they synchronize, they are only downloading data from the Mobile Server. The only difference, then, between the clients is when they synchronize to download the data. Since the Mobile clients could synchronize at different times, each are provided a unique state, where the cached user and state combination informs the server how much data to download to each particular Mobile client.

To use this feature, perform the following:

1. Ensure that the publication items are read-only. When adding publication items to the publication, pass the value of `IUD` for the argument `DISABLED_DML`. In this state, client-side DML modifications cannot be uploaded to the Mobile Server.
2. Create a single user and subscribe it to the read-only publication. For each group of clients that subscribe to the same subset of data, create one user. In the simplest case, where all clients share the same data, create a single user and subscribe it to the read-only publications.
3. Configure the users that are sharing the data, as follows:
   a. On Mobile Manager, navigate as follows: Data Synchronization->Administration->Instance Parameters.
   b. Configure each user that is to share the data from the read-only publications in the `CACHED_USERS` field. If you have more than one cached user, separate each user by a comma.

> **Note:** In this scenario, you may consider using Offline Instantiation for installing the Mobile client and the initial data download on each client. See Chapter 9, "Offline Instantiation" for more information.

A restriction for this feature is that the synchronization user can only subscribe to read-only publication items. A workaround is to create a shared user for all of the clients and a unique user for each client. The shared user subscribes to the shared read-only publication items and the unique user subscribes to the updateable publication items. The Mobile client needs to synchronize both users.

5.7 How Do You Encrypt All Databases for the Initial Sync?

In the default server configuration, Mobile clients do not automatically encrypt the snapshot ODB files after you complete the initial sync. However, you can modify your configuration to automatically encrypt all snapshot ODB files with the synchronization user password after the initial sync completes. The following sections demonstrate how to configure for this option either on the client or on the server:

- Section 5.7.1, "Configuring on the Local Client for Automatic Encryption of Local Snapshots"
- Section 5.7.2, "Configuring on the Server for Automatic Encryption of Local Snapshots"

5.7.1 Configuring on the Local Client for Automatic Encryption of Local Snapshots

On the local client, you can configure for automatic encryption of the snapshot ODB files after initial synchronization by modifying the `POLITE.INI/POLITE.TXT` file by setting the `ENCRYPT_DB` parameter in the `SYNC` section.

For details on what value to use for the `ENCRYPT_DB` parameter in the `POLITE.INI/POLITE.TXT` file, see Section F.3.2.12, "ENCRYPT_DB".

5.7.2 Configuring on the Server for Automatic Encryption of Local Snapshots

On the server, you can configure for automatic encryption of the snapshot ODB files after initial synchronization by performing the following:

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. Click **Create Command**.
5. Create the following new Command:

```
Name: EncryptDB
Command: updt_conf.otl
Description: Encrypt Database
```

6. Edit the newly created command `EncryptDB`, as follows:

```
Command: updt_conf?app=sync&key=ENCRYPT_DB&val=1
```

7. Apply the changes.
8. Edit the `DeviceInfo` Command. Insert the new Command `EncryptDB` and click **OK**.

For more information on sending commands to the Mobile device, see Section 7.7, "Sending Commands to Your Mobile Devices".

5.8 Managing Trace Settings and Trace Files

You can configure the type of tracing that occurs for Data Synchronization components. For more information, see See Section 3.1.2, "Data Synchronization Tracing" in the *Oracle Database Lite Troubleshooting and Tuning Guide*.

5.9 Browsing the Repository for Synchronization Details

The Repository screen describes how to look up user information, publications, publication items, and the In-Queue, Out-Queue, and Error queues that facilitate synchronization. This section contains the following topics:

- Section 5.9.1, "Viewing User Information"
- Section 5.9.2, "Viewing Publications"

- Section 5.9.3, "Viewing Publication Items"
- Section 5.9.4, "Viewing Synchronization Queues"

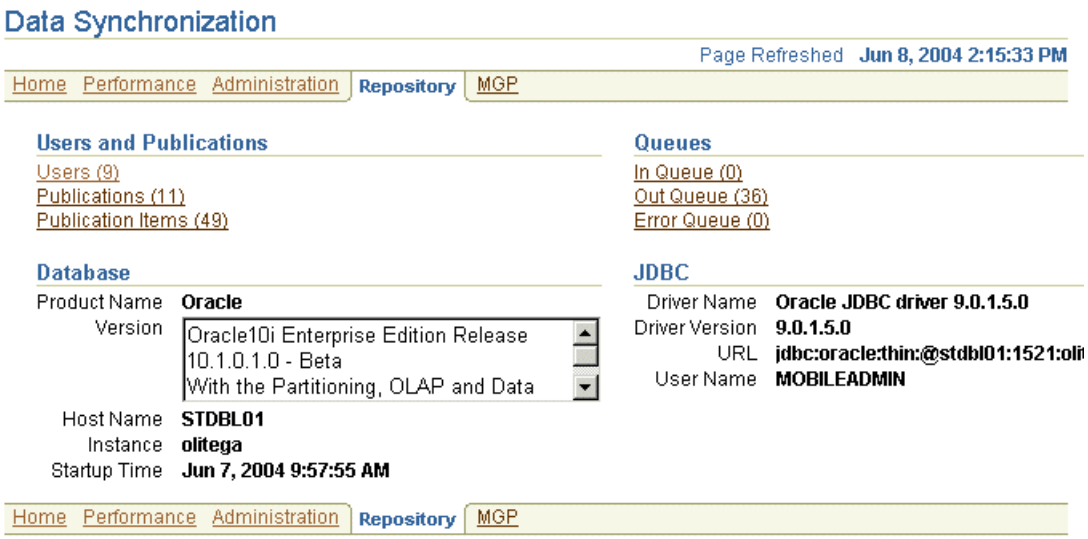
5.9.1 Viewing User Information

All users that have been added by the administrator (see Section 4.1, "Managing Users and Groups") are contained within the Mobile repository. With this Users screen, you can view everything that is attached to this user, such as application subscriptions, publication items, parameters, SQL scripts, Java resources, sequences, and performance analysis.

1. To view information about existing users in the repository, click the **Repository** tab on the Data Synchronization home page.

   As displayed in Figure 5–12, the Repository tab appears.

Figure 5–12 Repository Tab



2. Click **Users**, which brings up a list of all users currently in the repository. The number next to Users details the number of users currently in the repository.
3. Choose the user in which you are interested and click **Subscriptions**. The Subscriptions page displays the existing publications for the user. A subscription is the combination of the publication, its publications items, and the user to which it is attached.

   On the subscriptions screen, choose any publication and then click any of the buttons above it to see all of the publication items, parameters, SQL scripts, Java resources, and sequences. In addition, if you click the Consperf performance analysis button, you can generate performance analysis for the publication items. See Section 5.10.3, "Analyzing Performance of Publications With the Consperf Utility" for more information on Consperf performance analysis.

You can add subscriptions to the user by granting the user access to the application that contains the publication. See Section 4.2.1, "Grant or Revoke Application Access to Users" on how to grant access to applications. To add a publication to an application, use the Mobile Database Workbench.

5.9.2 Viewing Publications

To view all publications that have been published against the Mobile Server, click **Publications** under the Users and Publications section. The number next to Publications are the number of publications currently in the repository, which were uploaded to the repository when the application was published. You can view these publications individually using this link. Clicking **Publications** brings up a screen that contains a list of all of the publications. If there are too many to fit on a page, you can search for a specific publication. Similar to the Users screen, you can select a publication and then view the publication items, parameters, SQL scripts, Java resources, sequences, and users that are attached to this publication.

When you add publication items to each publication, you specify certain properties for each publication item within the publication, such as the order weight of when this item is executed in relation to the other publication items in the subscription, who wins when a conflict occurs, and options for disabling DML. You can view some of these properties when you select the publication and click **Publication Items**. For more information on these properties, see Section 3.3.1.6 "Adding Publication Items to a Publication" in the *Oracle Database Lite Developer's Guide*.

You can also view the users subscribed to each publication by selecting the publication and then clicking **Users**. When you do so, you see the following information about each user subscribed:

- User: the user name
- Parameter Set: If a data subsetting parameter is required for the publication, then this indicates if the parameter was set for this user.
- Instantiated: This indicates if the user is ready for synchronization. If there is a data subsetting parameter for the publication, but the value has not been set for this user, then this value is NO. The value is YES if either there is no data subsetting parameter or that the required parameters are set.
- Complete Refresh Requested: Indicates if this user requests a complete refresh.

You can only view publications in this screen. To modify your publication, use the Mobile Database Workbench. For more information, see Chapter 5 "Using Mobile Database Workbench to Create Publications" in the *Oracle Database Lite Developer's Guide*.

5.9.3 Viewing Publication Items

To view publication items, click **Publication Items** under the Users and Publications section. The number next to Publication Items details the number of publication items stored in the repository. These items were uploaded to the repository when the application was published.

Click **Show** to view the publication item properties.

You can only view publication items in this screen. To modify you publication and its publication item, use the Mobile Database Workbench. For more information, see Chapter 5 "Using Mobile Database Workbench to Create Publications" in the *Oracle Database Lite Developer's Guide*.

5.9.4 Viewing Synchronization Queues

You can view what is currently in the synchronization queues. To view transactions that are listed in queues, click the required hyperlink under the Queues section. For example, to view transactions that are listed in the Out-Queue, click **Out Queue**. The

number next to each queue shows the number of transactions contained within that queue.

The In-Queue and Error Queue are organized by transactions.

- Section 5.9.4.1, "Viewing Transactions in the In-Queue"
- Section 5.9.4.2, "Viewing Subscriptions in the Out-Queue"
- Section 5.9.4.3, "Viewing Transactions in the Error Queue"

5.9.4.1 Viewing Transactions in the In-Queue

You can view the current transactions that exist in the In-Queue. If you are wondering if your changes have been applied to the application tables, you can verify if they are still in the In-Queue or have already been processed by the MGP. If you see your transactions held in the In-Queue longer than you wish, then modify the timing on how often the MGP executes in the Job Scheduler. See Section 6.3, "Manage Scheduled Jobs Using the Mobile Manager" for more information on the Job Scheduler.

5.9.4.2 Viewing Subscriptions in the Out-Queue

The Out-Queue contains the transactions that are destined for the Mobile client. The transactions are organized by subscriptions, which is a combination of the user and each publication for the user. Also, you can see if a complete refresh is requested. Figure 5–13 displays the Out-Queue Publications page.

Figure 5–13 Out-Queue Publications Page

Out Queue Publications

Page Refreshed Sep 8, 2004 6:18:03 PM

Search User Go

Publication Items

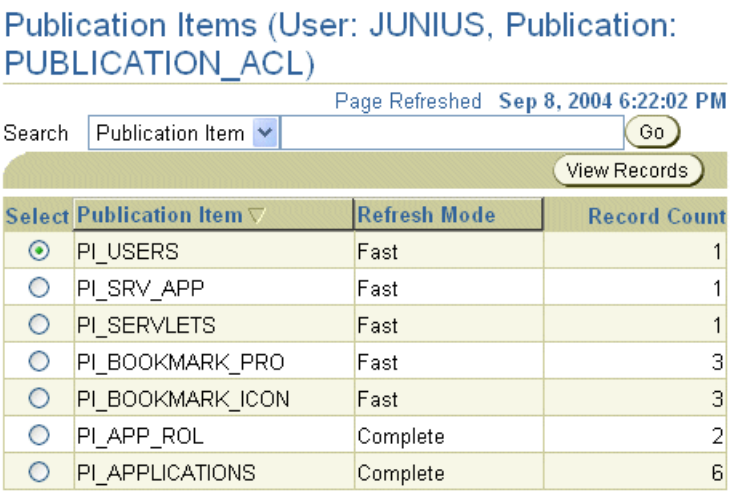
Select	User	Publication	Complete Refresh Requested
◉	S11U1	DEFAULT	Yes
○	S11U1	PUBLICATION_ACL	No
○	S11U1	WEBTOGO	No
○	JUNIUS	DEFAULT	Yes
○	JUNIUS	PUBLICATION_ACL	No
○	JUNIUS	WEBTOGO	No
○	JOHN	DEFAULT	Yes
○	JOHN	PUBLICATION_ACL	No
○	JOHN	WEBTOGO	No
○	JANE	DEFAULT	Yes
○	JANE	PUBLICATION_ACL	No
○	JANE	WEBTOGO	No
○	JACK	DEFAULT	Yes
○	JACK	PUBLICATION_ACL	No
○	JACK	WEBTOGO	No

You can view the details of each subscription by performing the following:

1. Select the subscription to view with the Select button next to the user name/publication in which you are interested.
2. Click **Publication Items**, which brings up Figure 5–13.

The Publications Items screen describes how many records is in the publication and whether it uses a fast or complete refresh mode.

Figure 5–14 Publication Items in the Out-Queue Subscription



3. View the records of the publication item by clicking the Select button and then click **View Records**.
4. Click **Show** on each record to see the record data.

5.9.4.3 Viewing Transactions in the Error Queue

The purpose of the error queue is to store transactions that fail due to errors or conflicts that can arise when a client synchronization does not perform as planned. Synchronization errors cause a rollback of the application of the client data to the server database and the error is posted to the error queue. In order to have the transaction apply to the base tables in the server database, you must resolve the error condition and re-apply the transaction.

Synchronization conflicts are resolved by the MGP using the relevant conflict rules. All modifications are applied to the server tables and the transaction is committed. If the conflict rule is "Server Wins", then an error queue record with the message `CONFLICT DETECTED` is also generated to let you know that a conflict occurred and this rule was applied. A conflict that is resolved by the conflict rules is not rolled back. You can choose to override the conflict resolution performed by modifying the error queue record for a conflict and re-executing the transaction.

A Mobile Server synchronization conflict occurs if:

- The client and the server update the same row. This error is resolved by the Mobile Server by the conflict rules, but is logged in the error queue for you to see the result. You can choose to modify the result.
- The client and server create rows with the same primary key values. This error is resolved by the Mobile Server, but is logged in the error queue for you to see the result. You can choose to modify the result.
- The client deletes the same row that the server updates. This error is resolved by the Mobile Server, but is logged in the error queue for you to see the result. You can choose to modify the result.

A Mobile Server synchronization error occurs if:

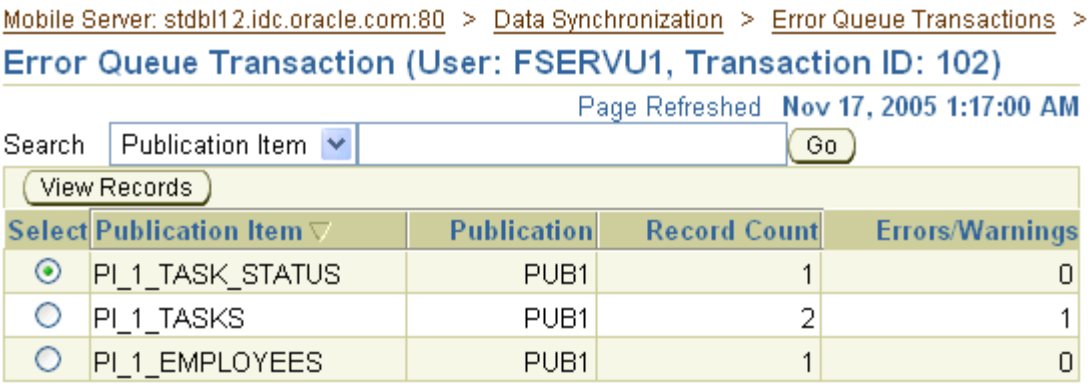
- The server deletes the same row that the client updates. This error is unresolved by the Mobile Server. The administrator must decide how this is resolved.
- Client is out of sync. This error is unresolved by the Mobile Server. The administrator must decide how this is resolved.
- Client records violate server database constraints. This error is unresolved by the Mobile Server. The administrator must decide how this is resolved by either modifying the database constraints and re-executing the transaction, or by modifying the error queue record to conform to the constraints.
- An error occurs when reapplying a backup. See Section 4.3, "Oracle Database Lite Backup Coordination Between Client and Server" in the *Oracle Database Lite Troubleshooting and Tuning Guide* for instructions on recovering from a backup.
- An unexpected error occurs with the back-end database, such as a constraint violation or storage issue.

> **Note:** Normally, only the first error is reported if an error occurs in the apply phase of the transaction. If you want to view all errors that occur for the transaction, set the `REPORT_ALL_ERRORS` parameter to `YES` in the Consolidator parameters, which is set in the Instance Parameters section of the Mobile Manager GUI.

If the administrator resolves the error condition that caused the problem, then the administrator may attempt to re-apply the transaction or purge the error queues. If you decide to reapply the records in the transaction to the application tables, perform the following:

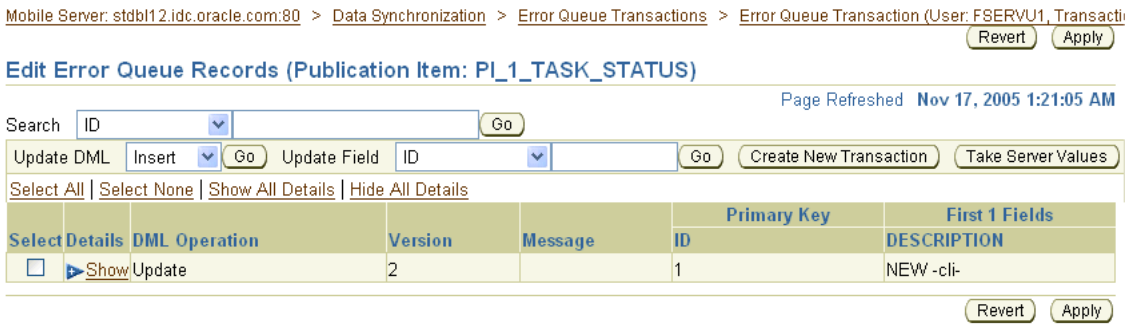
1. To view the transaction details, select a particular transaction and click **Publication Items**. This shows the Error Queue transaction with the associated publication items with a number designating Sync Errors or Conflicts. If a publication item shows an error count of zero, no errors have been reported.

Figure 5–15 Publication Items in the Error Queue



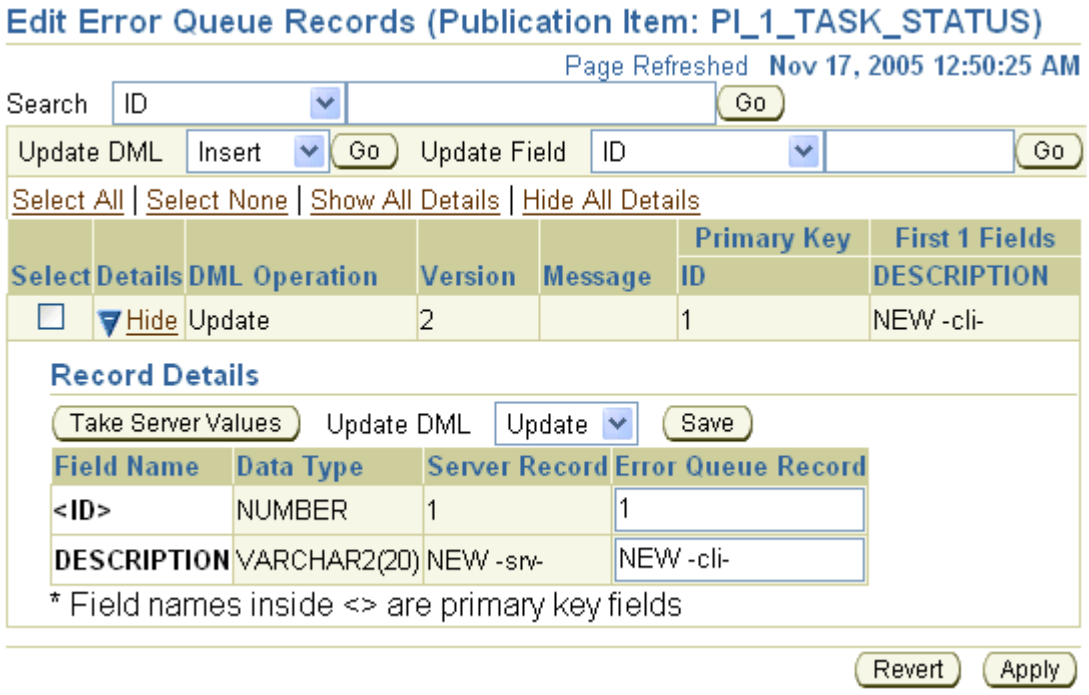
2. Select the publication item to analyze and correct, and click **View Records**, which brings you to the Edit Error Queue Records page, as shown in Figure 5–16. All of the details of the client and server data are displayed. If the Message field for any row is not blank, it means that this record produced a Sync Error or a Conflict. A message of `CONFLICT DETECTED` signifies a Sync Conflict; any other Message indicates a `Sync Error`. A blank Message field implies that the record executed correctly.

Figure 5–16 Error Queue Records for a Single Publication Item



The Error Queue record displays the data that arrived from the client and the data on the server for the corresponding row, if any. If you click **Show** under Details, then you can edit the Error Queue record, as shown in Figure 5–17.

Figure 5–17 Show Details of Error Queue Record



3. Correct the reason why the error occurred in the first place by either modifying the record details or by taking the server-side values.
   - Edit fields—Modify any fields in the Error Queue Record section under Record Details and click **Save**.
   - Update DML—You can only re-execute the transaction if the DML operation is changed to an Insert, Update, or Delete DML operation. If the DML operation is in the Error state, then modify the DML operation from Error to Update. Select a value from the Update DML drop-down list under Record Details and click **Save**.
   - Take Server Values—Click **Take Server Values** under Record Details. If a server record is present, the server-side values will appear in Error Queue

Record. That is, the Error Queue Record will have the same values as Server Record.

Note: If the conflict resolution is set to "server wins," then you may lose the client modifications. Thus, if you set the conflict resolution to "client wins," then you force these changes to overwrite the server.

---

4. After completing one or more of the above operations, click **Apply** to commit the changes. Clicking on **Revert** causes the changes to be cancelled. Avoid navigating to some other page without first clicking **Apply** or **Revert**.

- Alternatively, you can update several transactions at once on a group of selected records; that is, the operations are simultaneously executed for all records for which the Select checkbox is checked. After selecting the desired rows, choose the buttons at the top of the page ot perform one or more of the following:
- Update DML—Select a set of rows using the checkboxes. Choose a value from the Update DML drop-down list at the top of the page and click **Go**.
- Update Field—Select a set of rows using the checkboxes. Choose the name of the field from the Update Field drop-down list which contains all the editable fields in the table. Then enter the value you want to enter for that field and click **Go**.
- Take Server Values—Select a set of rows using the checkboxes and click on Take Server Values at the top of the page.
- Create New Transaction—Select a set of rows using the checkboxes and click on Create New Transaction. The selected records will be removed from the current Error Queue transaction and a fresh transaction is created that consists of only these records. This operation may be used when you want to re-apply on a certain set of the original records. The new transaction could either be purged to applied separately.

After completing one or more of the above operations, click **Apply** to commit the changes. Clicking **Revert** causes the changes to be cancelled. Avoid navigating to some other page without clicking **Apply** or **Revert**. You do not need to click **Apply** or **Revert** for the Create New Transaction operation.

5.10 Monitoring and Analyzing Performance

The following sections describe how to monitor and analyze Data Synchronization performance.

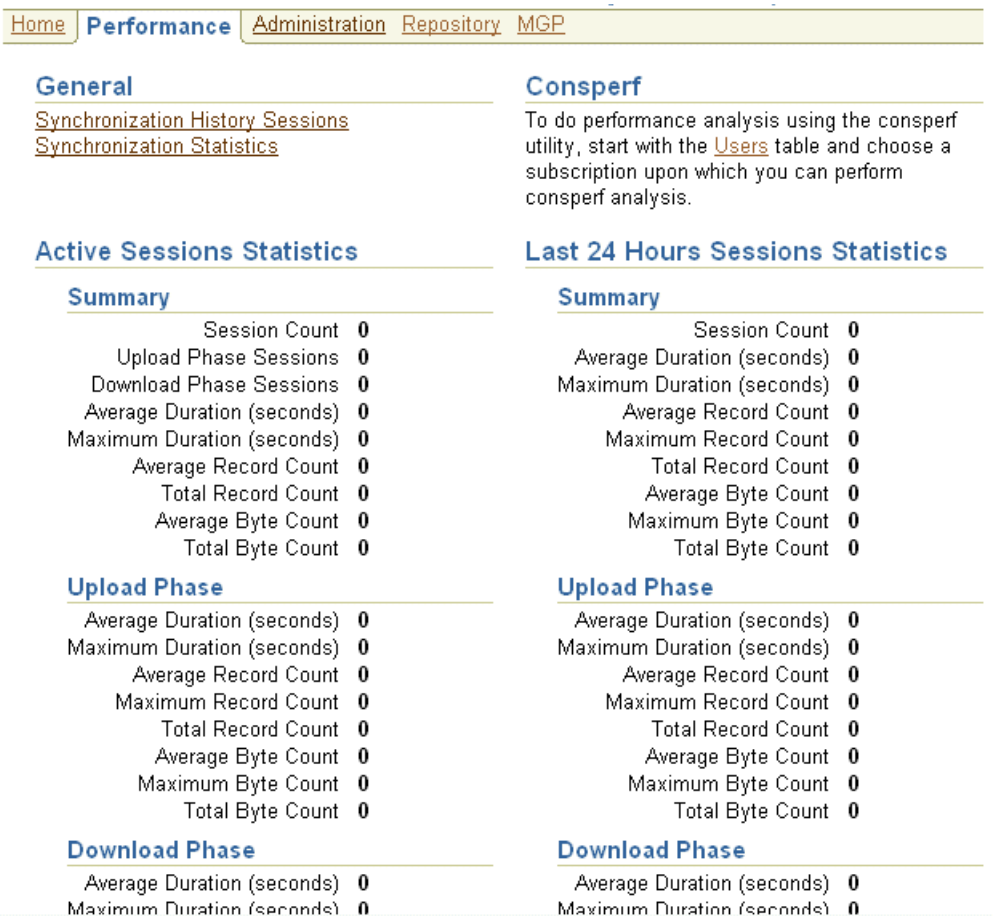
5.10.1 Viewing Sync Server Statistics

The Performance tab displays the Sync Server statistics of the current session and statistics of history sessions that have occurred in the last 24 hours.

To view Sync Statistics, click the **Performance** tab. As displayed in Figure 5–18, you can see the active Sync Server statistics from the currently active sessions and compare

it to overall statistics gathered from all sessions in the past 24 hours. This includes an overall section, the upload phase, and the download phase.

Figure 5–18 Performance Page

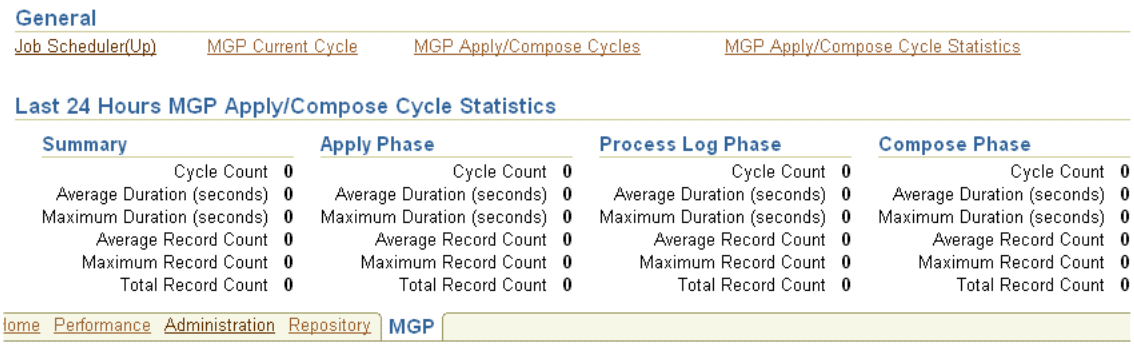


To view statistics from other dates, click the **Synchronization Statistics** link in the General section of this page. The Synchronization Statistics page contains search criteria such as user name, device type, and duration. Specify your criteria in the Search section and click **Go**. The Sync Statistics page displays results such as summary, upload phase, and download phase details.

5.10.2 Viewing MGP Cycles and Statistics

By navigating to the MGP tab off of the Data Synchronization screen, the Mobile Server administrator can view the current status of the MGP cycle (see Figure 5–19). The columns are separated so that you can see how, in the last 24 hours, the MGP has performed overall, as well as for each individual phase: apply, compose and process.

Figure 5–19 MGP Page



When you click on MGP Current Cycle, you can see what the MGP process is currently doing. For instance, you can check if the apply or compose cycle is running when the MGP cycle is in progress. If you have set the `MGP_HISTORY` instance parameter, (see Section 5.5, "Configuring Data Synchronization For Farm or Single Mobile Server"), then upon completion of the apply or compose cycle, the cycle details are stored in Cycle History.

Since the front page only shows the last 24 hours, you can view farther back by clicking on the MGP Apply/Compose Cycle Statistics. You can set a date range to search and can even specify whether to search based upon the following:

- Apply Only or Apply/Compose
- Success, Failure, or Conflict results

When you click MGP Apply/Compose cycles, you can search for a range of historical records of these cycles and then view the details of each cycle.

5.10.3 Analyzing Performance of Publications With the Consperf Utility

The Consperf utility profiles your subscriptions and may modify how the publication item is executed if the utility determines that there is a more performant option. The Consperf tool evaluates how the SQL within the publication item interacts with our Data Synchronization query templates. The first synchronization is always a complete refresh, which is a direct invocation of the query. On subsequent synchronizations, the query templates determine incremental refreshes. This improves your performance from not having to perform a complete refresh each time you synchronize. However, the interaction of our query templates and your SQL may not be optimal, which is discovered by the Consperf tool. We either modify the query template or type of logical delete or insert for you or you can adjust your SQL to be more performant in regards to our templates.

In addition, application developers and administrators use this utility to analyze the performance of subscriptions and identify potential bottlenecks during synchronization.

This tool generates the following two primary analysis reports:

1. Timing statistics for publication items
2. Explain plans for publications

The Consperf tool automatically tunes subscription properties, if the default templates do not supply the highest performing option. You can select a client and choose the desired subscription for performance analysis. Users can change parameter values before analyzing performance. The analysis results, which are timing and execution

plan reports, are stored on the server and can be accessed by viewing the same user and subscription.

For a full description of how to use the Consperf utility, see Section 1.2.1 "Analyzing Performance of Publications With the Consperf Utility" or for the Explain Plan, see Section 1.4, ""Determining Performance of Client SQL Queries With the EXPLAIN PLAN" in the *Oracle Database Lite Troubleshooting and Tuning Guide*.

5.10.4 Monitoring Synchronization Using SQL Scripts

If, instead of viewing MGP statistics within the Mobile Manager, you can execute SQL scripts to monitor Mobile application status during synchronization. For a full description of how to monitor synchronization, see Section 1.2.2. "Monitoring Synchronization Using SQL Scripts" in the *Oracle Database Lite Troubleshooting and Tuning Guide*.

6

Job Scheduler

The following sections describe how you can manage the job engine and schedule jobs through the Mobile Manager.

- Section 6.1, "Scheduling a Job to Execute at a Specific Time or Interval"
- Section 6.2, "Managing the Job Engine"
- Section 6.3, "Manage Scheduled Jobs Using the Mobile Manager"
- Section 6.4, "Managing Scheduled Jobs Using ConsolidatorManager APIs"
- Section 6.5, "Using the ConsolidatorManager APIs to Create Jobs"

6.1 Scheduling a Job to Execute at a Specific Time or Interval

You can choose to execute any job—which can be any application—at a specific time or interval. For example, the default MGP process (see Section 5.1, "How Does the Synchronization Process Work?") is a job that executes at a regular interval. The default behavior is for the MGP process to execute every 60 seconds to apply all incoming modifications from the clients and compose all outgoing messages to the clients from the repository. You can define how often the MGP process executes, or even schedule a time for it to stop execution. You can schedule any job with this same mindset.

> **Note:** For an overview on how to create a job out of one of your applications, see Section 6.5, "Using the ConsolidatorManager APIs to Create Jobs".

The engine that monitors the job execution is the Job Scheduler. For example, by default, the Job Scheduler fires off the MGP process every 60 seconds. It is the mechanism that tracks all of the scheduled jobs and ensures that your defined job is executed when you wanted it to be executed. You can turn it on and off, and monitor alerts specific to the Job Scheduler.

See Section 6.2, "Managing the Job Engine" for details on how to manage the job engine; see Section 6.3, "Manage Scheduled Jobs Using the Mobile Manager" on how to create and manage jobs that you want scheduled.

> **Note:** Within the OC4J or Web-to-Go client, you can also schedule when a synchronization is started on the client. This is separate from the Mobile Server Job Scheduler. See Section 2.3.1.1.2, "Configuration Tab" for more information.

6.2 Managing the Job Engine

The Job Scheduler manages jobs that you create and schedule. However, the Job Scheduler needs to be managed, as well. You navigate to the Job Scheduler page in the Mobile Manager by selecting a Mobile Server from the list of Mobile Servers on the Mobile farm screen. At the bottom of the Mobile Server screen, click **Job Scheduler**, which brings up the Home screen for the Job Scheduler.

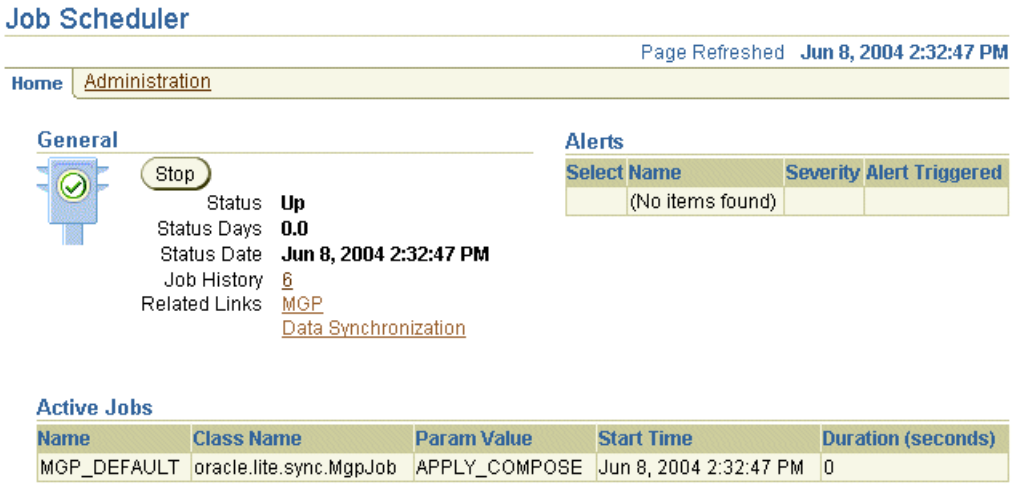
The following sections describe how to manage the Job Scheduler:

- Section 6.2.1, "Starting the Job Scheduler"
- Section 6.2.2, "Checking Job Scheduler Alerts"
- Section 6.2.3, "Managing Active Jobs"
- Section 6.2.4, "Managing the Job History List"

6.2.1 Starting the Job Scheduler

Figure 6–1 displays the Job Scheduler's default status on the Job Scheduler home page. To start the Job Scheduler, click **Start**. At this stage, the "Start" button is replaced by the "Stop" button. The following image displays that the Job Scheduler is up and running.

Figure 6–1 The Job Scheduler Home Page



To stop the Job Scheduler, click **Stop**. Stopping the Job Scheduler prevents any new jobs from starting. However, any existing jobs will continue to execute until finished. Stopping the Job Scheduler does not kill any existing jobs. If you want to prevent only a single job from being launched, then disable the application on the Administration screen. See Section 6.3.3, "Enabling or Disabling Jobs" for more information on disabling applications.

6.2.2 Checking Job Scheduler Alerts

When the Job engine fails, then the Alerts table displays these exceptions as critical alerts. When the Job engine has trouble with executing your job, then these exceptions are displayed as warning alerts.

The Job Scheduler home page enables administrators to check alerts that are registered in the job engine. To check alerts, locate the "Alerts" table and select the alert that you need to view under the **Select** column. Click **Check**.

6.2.3 Managing Active Jobs

As shown in Figure 6–1, the Active Jobs table on the Job Scheduler home page contains information—such as job name, class name, parameter value, job start time, and duration.

For more information on how to manage jobs, see Section 6.3, "Manage Scheduled Jobs Using the Mobile Manager".

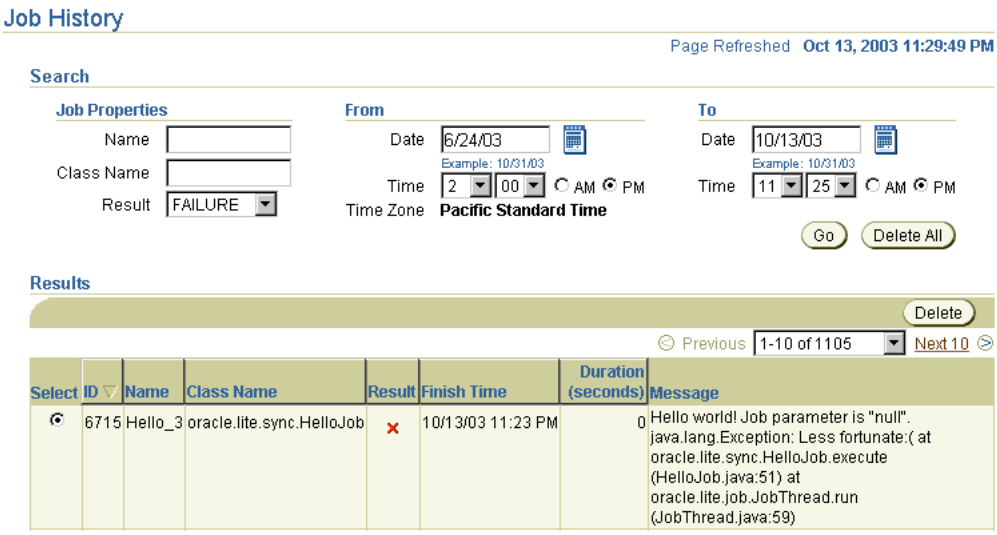
To terminate a job, click the **Administration** tab, select the job, and click **Delete**. This does not terminate active jobs, but prevents the job from executing in the future.

6.2.4 Managing the Job History List

The number of current registered jobs in the Job History list is listed on the Job Scheduler home page under the Status Date line. Click on the number displayed to bring up the Job History page. The Job History page enables you to provide criteria to search, sort, and manage the job history based on job properties—such as name, class name, result, or a specific date and time. Based on your search criteria, the Job History page displays job history details under the **Results** section.

Figure 6–2 displays the Job History page.

Figure 6–2 Job History Page



You can sort the messages by any of the headers. For example, to sort the job history details by name, click **Name** in the header title region. It toggles between A-Z and Z-A.

To delete a single job, select the job and click **Delete**. To delete all job history entries that match your search criteria, click **Search and Delete**.

6.3 Manage Scheduled Jobs Using the Mobile Manager

The most notable scheduled job is the MGP (see Section 5.1, "How Does the Synchronization Process Work?"). By default, it is scheduled to execute every 60 seconds to perform a specific task for data synchronization. You can modify the schedule of this existing job, as well as create other jobs for your own purposes to execute at a regular interval.

From the Job Scheduler screen, click the Administration tab, where you can create a new job or edit existing jobs. The Scheduled Jobs section displays the jobs that are scheduled in the job engine.

The following sections enable administrators to accomplish the following tasks:

- Section 6.3.1, "Creating a New Job"
- Section 6.3.2, "Editing Existing Jobs"
- Section 6.3.3, "Enabling or Disabling Jobs"
- Section 6.3.4, "Deleting Jobs"
- Section 6.3.5, "Default Jobs"

6.3.1 Creating a New Job

In order to create a new job, you create the schedule of how often and when an existing application is executed. To create this schedule, navigate to the Job Scheduler Administration screen and click **Create A New Job**.

Figure 6–3 displays the top section of the Create a New Job page. Give the job a name, select the checkbox for Enabled to enable the job (or leave blank to leave disabled), and select the checkbox for Save to Job History if you want a record of this application executing.

Under the Job Class section, if this is for the MGP process, then select if this is for Apply Only or Apply and Compose. The MGP process can be modified to perform only application of new and modified records from the clients. This is beneficial for applications that never have to update information from the back-end server database. Choosing Apply Only saves performance if it is relevant for your application. For example, if you had a company that performed a lot of updates throughout the day, but no one needed to know the new information until the next day, you could schedule an MGP process to perform Apply Only all day to update the repository, and schedule another MGP process that executes only at night with Apply/Compose to perform the last updates and then bring down all of the days modifications to all of the users.

If this is not for an MGP process, then enter the class name to be executed for this job and any parameter values. Since you design the class, enter the parameter as you have designed the parameter format. There are two default jobs, which are described in Section 6.3.5, "Default Jobs".

Figure 6–3 Create a New Job - Top Section

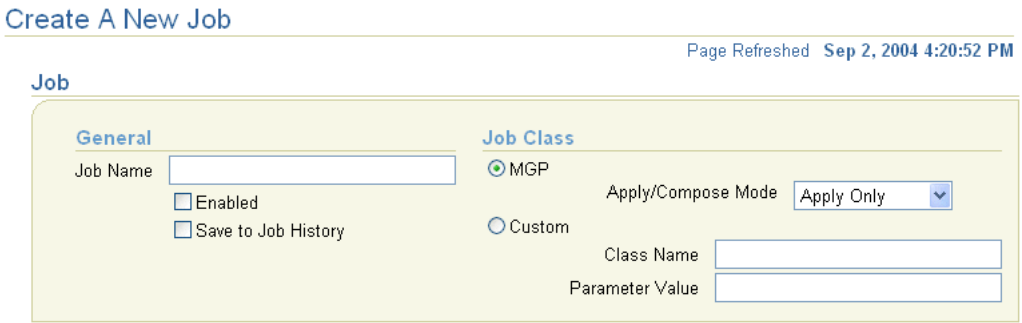
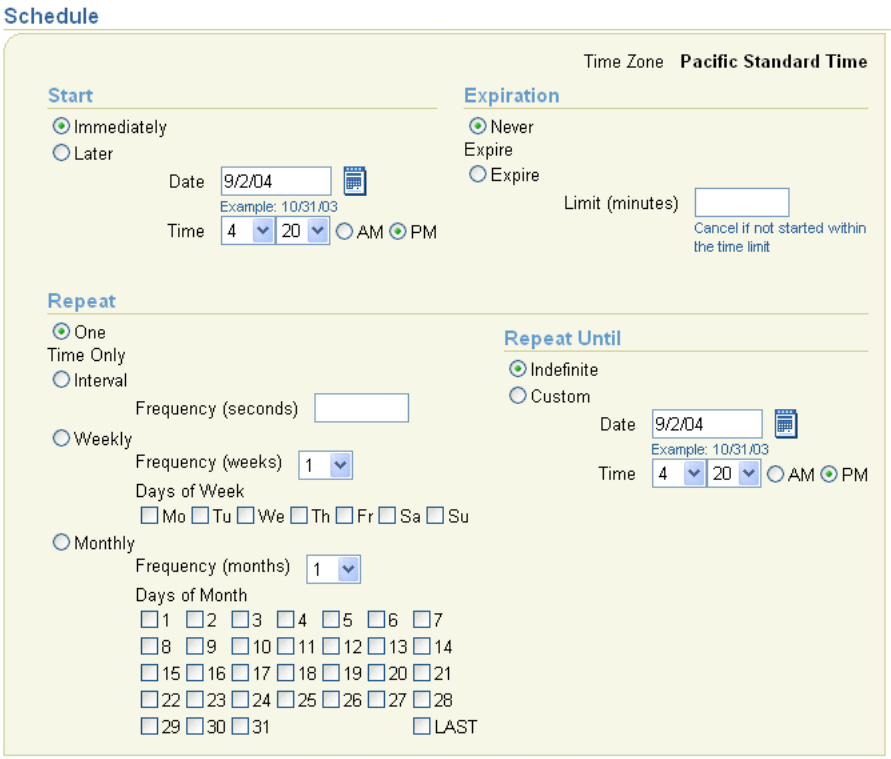


Figure 6–4 displays the bottom section of the Create a New Job page, which is where you define when and how often your job executes.

Figure 6–4 Create a New Job - Bottom Section



Enter data in the Create a New Job page as described in the following tables.

Table 6–1 describes data that must be entered in the **Start** section.

Table 6–1 Start Details Description - Schedule Section

Field	Description	Required
Immediately	To start the job immediately, select this option.	Optional
Later	To start the job at a later time, select this option and specify the date and time when this job is to start.	Optional

Table 6–2 describes data that must be entered in the **Expiration** section.

Table 6–2 Expiration Details Description - Schedule Section

Field	Description	Required
Never Expire	To ensure that the chosen job schedule does not expire—that is, this job always executes—select this option.	Optional
Expire	If you want the job to expire after a specified number of minutes—even if it has not execute yet—then specify the number of minutes in this field. The Job Scheduler cancels jobs that do not start at the specified time. However, it does not stop jobs that have already started.	Optional

Table 6–3 describes how often the job executes in the **Repeat** section.

Table 6–3 Repeat Details Description - Repeat Section

Field	Description	Required
One Time Only	The job executes only once.	Optional
Interval	The job executes after a specified interval has passed. The interval duration between execution of the job is defined in seconds.	Optional
Weekly	The job executes on the specified day of the week. You can specify an interval of whether this executes weekly (1 in the Frequency pulldown), every other week (2 in the Frequency pulldown) and so on.	Optional
Monthly	The job executes on a specified day of the month. Same as above, but with the months of the year.	Optional

Table 6–4 defines whether the chosen schedule repeats indefinitely or whether you want it to execute only on a certain date/time.

Table 6–4 Repeat Until Details Description - Repeat Section

Field	Description	Required
Indefinite	To repeat the job schedule indefinitely, select this option.	Optional
Custom	To specify how long this job executes until, specify the date and time of when the job is canceled.	Optional

To implement the job schedule after specifying changes to the schedule, click **OK**. To retain or restore previous job schedule values, click **Cancel**.

Note: The calendar does not display the selected date if the Java script feature in your browser, any pop up blocking tools, or search tools are installed and enabled.

6.3.2 Editing Existing Jobs

Navigate to the Job Scheduler Administration screen. To edit existing jobs, click **Edit**. Modify the same fields that are described in Section 6.3.1, "Creating a New Job".

6.3.3 Enabling or Disabling Jobs

You can enable or disable a job from the Administration screen off of the main Job Scheduler screen. Select the job that you need to modify and either click **Enable** or **Disable**. The **Status** column confirms the changed status.

6.3.4 Deleting Jobs

Navigate to the Job Scheduler Administration screen. To delete a job, select the job that you need to delete and click **Delete**. The Job Scheduler displays a warning message that seeks your confirmation to delete the chosen job. Click **Yes**. You will be returned to the Administration tab.

6.3.5 Default Jobs

The Oracle Database Lite 10*g* Edition contains default jobs. As a user, you can enable or disable these default jobs and edit or delete them. This edition contains the following default jobs.

- MGP Process: `MGP_DEFAULT`
- Purging History: `PURGE_HISTORY_DEFAULT`

6.3.5.1 MGP_DEFAULT

You have to have at least a single MGP process for apply/compose phase of the synchronization phase. The MGP_DEFAULT is this process. You can modify this process to be apply only, or you can modify when the MGP process is executed. You can create other MGP processes, if you wish.

job Name

`MGP_DEFAULT`

Job Class

`oracle.lite.sync.MgpJob`

Job Parameter Value

`APPLY_COMPOSE`

The parameter value must be a string of the value `APPLY_COMPOSE` or `APPLY_ONLY`. When scheduling or editing this parameter using the Job Scheduler's Edit Jobs page, you can choose the required parameter value from the Apply/Compose list.

6.3.5.2 PURGE_HISTORY_DEFAULT

In order to preserve disk space, the administrator wants to purge the history. This job is created for you to automatically purge the history at a selected interval. You can modify the interval or disable this job, if you wish. This section describes the job class, job parameter value and its corresponding description.

Job Name

`PURGE_HISTORY_DEFAULT`

Job Class

`oracle.lite.sync.PurgeHistoryJob`

Job Parameter Value

`History=Sync,MGP,Job;Days=7`

Since this Job is a customized class, the parameter is defined and parsed within the purge history class. The structure of this parameter is a string with two name/value pairs: what type of history to purge and for how many days. In this example, the history purged is for the Sync, MGP, and Job historical data. The history is purged for the last seven days. You can modify the number of days or add/delete the history logs that this applies to. The only options are Sync, MGP, or Job. For example, if you want every record that is 3 days old or more to be erased, modify the 7 to a 3.

6.4 Managing Scheduled Jobs Using ConsolidatorManager APIs

Application developers can define, submit, and manage jobs programmatically based on a pre-determined time and interval. For example, jobs can be scheduled to run repeatedly for a specified duration on any specified day or days of the week or month. Administrators can schedule jobs to run repeatedly for a specified number of months, weeks or specified days of the month or week.

The Job Scheduler provides an API for scheduling and running jobs using a job engine. It is a generic component which enables apply and compose functions for MGP, device manager jobs, and custom jobs.

- Using the class `oracle.lite.sync.ConsolidatorManager`, application developers can register or de-register a job class, create, drop, enable or disable a job, search, and delete a job execution log.
- Use other supporting classes, such as `Job`, `Schedule`, `ExecutionResult` and `ExecutionLog` in the `oracle.lite.sync.job` package to manage your scheduled jobs.

For more information on these classes and their methods, refer to the *Oracle Database Lite API JavaDoc.*

6.4.1 Start Job Scheduler In Separate JVM

If you want to execute the Job Scheduler in a different JVM from the Mobile Server, then perform the following:

1. Retrieve a connection to the database with the Consolidator Manager `openConnection` method. Pass in the Mobile Manager administrator username, password and optionally, the JDBC URL to the back-end Oracle database.
2. Create a new Job engine with the `JobEngine` class and start it with the `startUp` method. The Job engine executes in a separate thread, which you can terminate from the main thread.
3. Define how long the thread is to sleep between execution of all jobs.
4. Terminate the Job engine when you have completed all activities.

> **Note:** The following example demonstrates how to start up the Job engine in another thread from the Mobile Server. It executes all of the jobs that have been scheduled either through the API or through the Mobile Manager Job Scheduler screens, because the Job Scheduler retrieves the scheduled job information from the repository.

```
JobEngine JobEngine = new JobEngine();
JobEngine.startUp();
if (JobEngine.runnerThreadException != null){
  System.out.println("runnerThreadException:");
  JobEngine.runnerThreadException.printStackTrace();
}

Thread.currentThread().sleep(60*1000);

if (JobEngine.runnerThreadException != null){
  System.out.println("runnerThreadException:");
  JobEngine.runnerThreadException.printStackTrace();
}
```

```
JobEngine.shutDown();
```

6.5 Using the ConsolidatorManager APIs to Create Jobs

Within the `oracle.lite.sync.ConsolidatorManager` class, there are several APIs, which are documented fully in the *Oracle Database Lite API JavaDoc*, that enable you to create, register, and schedule your job.

While these methods are described fully in the JavaDoc, the following demonstrates the order in which you would execute the methods:

1. Create your job class by implementing the `oracle.lite.job.Job` interface. Implement the Job interface methods, as follows:
   - `init` method—This method is invoked by the Job Scheduler when the job is loaded.
   - `execute` method—This method is invoked by the Job Scheduler when the job is scheduled to execute. Put a call into your application within this method. The Job Scheduler passes in the input parameter that was provided when the job is created—either with the `createJob` method or within the Mobile Manager Job Scheduler screen. When finished, the `execute` method returns an object of class type `ExecutionResult` containing whether the job was a success or failure.
   - `destroy` method—This method is invoked after the job completes.
2. After you have created your job class, register it with the `registerJobClass` method.
3. Create the job in the Job Scheduler by executing the `createJob` method. One of the input parameters is an object of class type `Schedule`, which defines when the job is executed. There are also other management methods that correspond to the Mobile Manager GUI, such as `dropJob`, `enableJob`, and `disableJob`.
4. If you want to retrieve any logs, execute the `getJobExecutionLogs` method, which retrieves objects of `ExecutionLog` class.

7

Manage Your Devices

When you install your Mobile client software, the Mobile device manager client software is automatically installed and, in most cases, bootstrapped. Within the Mobile Manager, the administrator can send commands to remote devices. The next time that the device is available—either through wireless connection or synchronization—the command that you send will execute.

The following sections describe how to manage your devices:

7.1 Installing the Mobile Client Software on Your Mobile Device

The Mobile device software for your language and platform is installed when you install the Mobile client on your device. See Chapter 2, "Managing Your Mobile Client" for more information.

You can customize the installation for your Mobile clients by customizing the platform and the setup configuration files for the platform, as follows:

- Certain modifications can be made to the Mobile client configuration files before installation. See Section 7.2, "Configuring Mobile Clients Before Installation" for more information.

- Customize the platform to install additional binaries, applications, and other environment modifications. See Section 7.5, "Configuring and Customizing Your Mobile Device Platform" for more information.

Once installed, you can manage the client using the Oracle Database Lite Device Manager tool. See Section 7.9, "Using the Device Manager Agent (dmagent) on the Client" for more information.

7.2 Configuring Mobile Clients Before Installation

When you install the Mobile client on the device, a few configuration files are installed, such as the `webtogo.ora`, `polite.ini` and `odbc.ini` files. However, you can pre-configure some of the parameters destined for the client `webtogo.ora`, `polite.ini` and `odbc.ini` files using either of the following:

- Using the Mobile Manager: Navigate to the Mobile Devices->Administration->Configuration Management page, which enables you to modify the parameters, located in the INF file, corresponding to the Oracle Database Lite platform that is to be downloaded to the client.
- Edit the `<ini>` section of the INF file: To edit the INF file directly, see Section 7.11, "Installation Configuration (INF) File".

Note: The `polite.ini` and `odbc.ini` files are available in Windows under `%WINDIR%` and in Linux under `$OLITE_HOME/bin`. For the Linux platform, you must have write permissions on the directory where these are located to be able to modify them.

---

1. Navigate to the Mobile Device screen.
2. Click **Administration**.
3. Click **Configuration Management**.
4. The parameters destined for the client configuration files are initially set up in different INF files. You can modify some of the parameters in these files. However, this is a very sensitive configuration and should only be done if you fully understand the function of each parameter. Normally, the only time you modify these parameters is from direction from Oracle Support.

   Select the INF file from the File Name pull down and click **Show**. This enables you to modify the INI section in this particular INF file. All of the current assignments are displayed.

   For each INF file, the parameters in each INI section is displayed. You can only add name value pairs to existing sections.
5. To add name/value pairs to the existing sections, click **Add**. To modify a parameter, modify it and click **Apply**. To delete, select the configuration pair and click **Delete**.
6. To add more sections, you must modify the INF file directly. To modify or add items to the existing sections, you can click **Add**.
7. A screen is displayed asking for two strings: a name and a value. Enter these items and click **OK**.

The following sections describe each INF file:

- Section 7.2.1, "Modifying Device Management Parameters for Client Device"

- Section 7.2.2, "Modifying WEBTOGO Parameters for Client Device"
- Section 7.2.3, "Modifying Oracle Lite Win32 Parameters for Client Device"

7.2.1 Modifying Device Management Parameters for Client Device

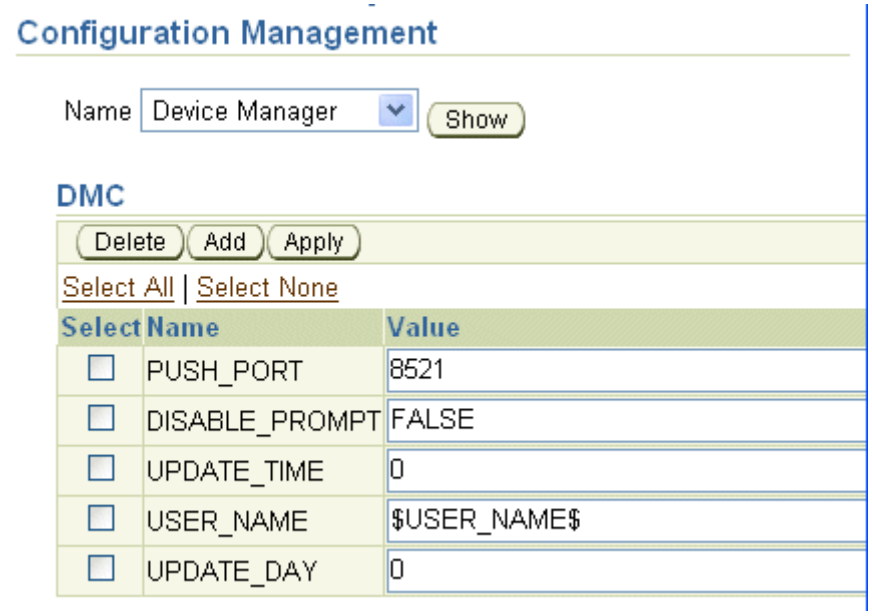
When you select Device Management from the File Name pull down, you can modify the `dmc.inf` file. The following example shows the INI section directly from the `dmc.inf` file:

```
<ini>
 <item name="POLITE" section="DMC">
 <item name='USER_NAME'>$USER_NAME$</item>
 <item name='PUSH_PORT'>8521</item>
 <item name='DISABLE_PROMPT'>FALSE</item>
 <item name='UPDATE_DAY'>0</item>
 <item name='UPDATE_TIME'>0</item>
 </item>
</ini>
```

Figure 7–1 displays how the Device Management INI section is displayed in the Mobile Manager. To add a name/value pair, click **Add**. To modify a parameter, modify it and click **Apply**. To delete, select the configuration pair and click **Delete**.

Note: Even though it is displayed here, you should not modify the `USER_NAME` field.

Figure 7–1 Adding Name/Value Pairs to Device Management INF File



Where the parameters are as follows:

Table 7–1 Device Management Parameters in DMC.INF File

Parameter	Description
PUSH_PORT	The listening port on the Mobile device for incoming commands from the Mobile Server. By default, the value is 8521. Port listed here is for all Mobile devices; thus, all clients are configured with the identical port number. Also, the server administrator can disable the PUSH_PORT completely (for security reasons) by setting the value of PUSH_PORT to zero. Do not modify the PUSH_PORT value on the client in the polite.ini file.
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.
UPDATE_DAY	Day when the Mobiledevice 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
UPDATE_TIME	Time of day that the Mobile 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
USER_NAME	Do not modify; automatically retrieves the username from the Mobile Server when downloaded to the client.

Note: You can also modify the UPDATE_DAY and UPDATE_TIME parameters on the client through the dmagent UI. See Section 7.9, "Using the Device Manager Agent (dmagent) on the Client" for details.

7.2.2 Modifying WEBTOGO Parameters for Client Device

When you select Web-to-Go from the File Name pull down, you can modify the `webtogo.inf` file that will be installed on the Mobile devices. The following example shows the INI section directly from the `webtogo.inf` file:

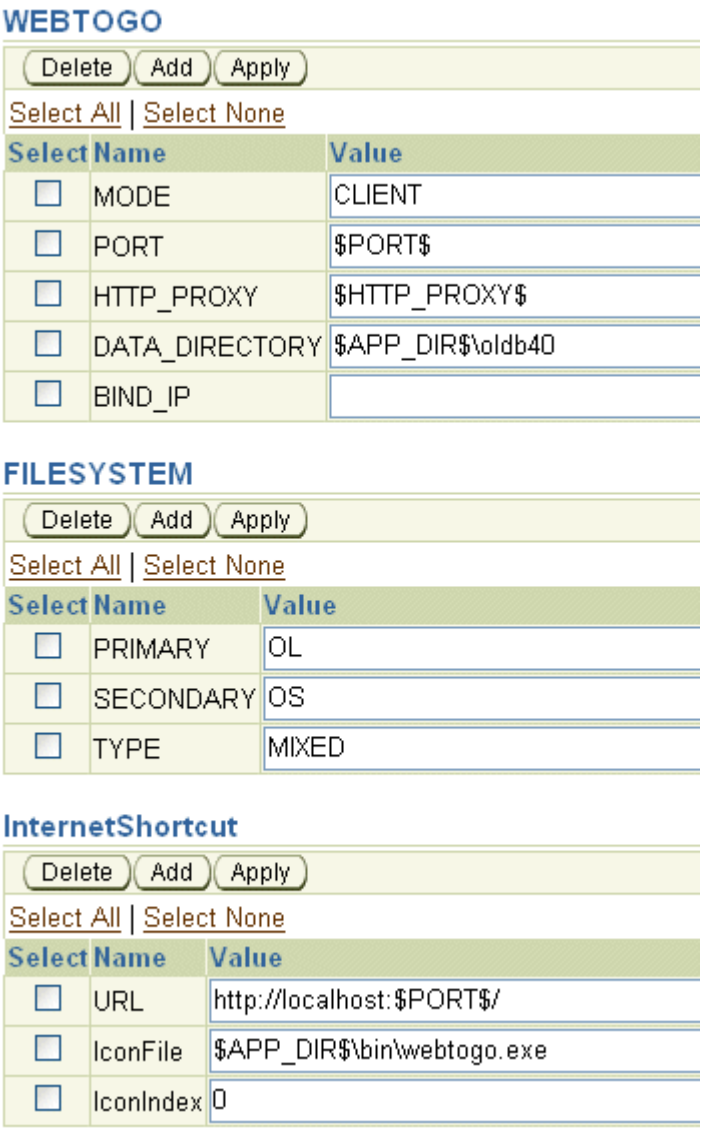
```
<ini>
  <item name="$APP_DIR$\bin\webtogo.ora" section="WEBTOGO">
    <item name="MODE">CLIENT</item>
    <item name="Data_Directory">$APP_DIR$\oldb40</item>
    <item name="PORT" replace="false">$PORT$</item>
    <item name="HTTP_PROXY" replace="false">$HTTP_PROXY$</item>
  </item>
  <item name="$DESKTOP$\Web-to-Go.url" section="InternetShortcut">
     <item name="URL" replace="false">http://localhost:$PORT$/</item>
     <item name="IconFile">$APP_DIR$\bin\webtogo.exe</item>
     <item name="IconIndex">0</item>
  </item>
</ini>
```

> **Note:** Most of the parameters in this section should only be modified if directed by Oracle Support.

There are two sections shown in this INI section: `WEBTOGO` and `InternetShortcut`. You can add other sections. Once added, the Mobile Manager reads them in and displays them accordingly.

Figure 7–2 displays how the Web-to-Go INI section is displayed in the Mobile Manager. To add a name/value pair, click **Add**. To modify a parameter, modify it and click **Apply**. To delete, select the configuration pair and click **Delete**.

Figure 7–2 Modifying Web-to-Go INF File Parameters



7.2.3 Modifying Oracle Lite Win32 Parameters for Client Device

When you select Oracle Lite Win32 from the File Name pull down, you can modify the `webtogo.inf` file. The following example shows the INI section directly from the `webtogo.inf` file:

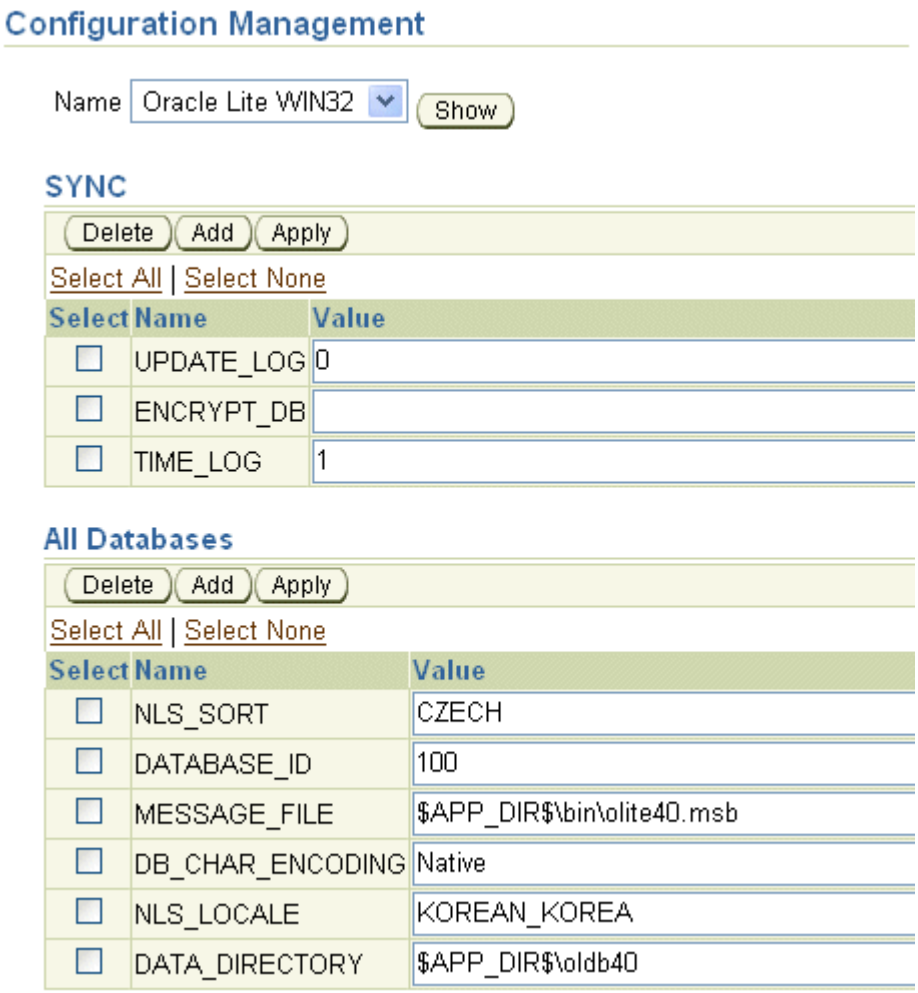
```
<ini>
  <item name="$APP_DIR$\bin\webtogo.ora" section="WEBTOGO">
    <item name="MODE">CLIENT</item>
    <item name="Data_Directory">$APP_DIR$\oldb40</item>
    <item name="PORT" replace="false">$PORT$</item>
    <item name="HTTP_PROXY" replace="false">$HTTP_PROXY$</item>
  </item>
  <item name="$DESKTOP$\Web-to-Go.url" section="InternetShortcut">
     <item name="URL" replace="false">http://localhost:$PORT$/</item>
     <item name="IconFile">$APP_DIR$\bin\webtogo.exe</item>
```

```
    <item name="IconIndex">0</item>
  </item>
</ini>
```

There are two sections configured in this INI section: `WEBTOGO` and `InternetShortcut`. You can add other sections. The Mobile Manager notes if there are sub-items and will create a heading for the sections.

Figure 7–3 displays how the Web-to-Go INI section is displayed in the Mobile Manager. To add a name/value pair, click **Add**. To modify a parameter, modify it and click **Apply**. To delete, select the configuration pair and click **Delete**.

Figure 7–3 Modifying Win32 INF File Parameters



7.3 Supporting Network Roaming for Devices With Broadbeam

Normally, Oracle Database Lite associates devices with a single network provider for Device Manager. The Mobile Server sends commands to the device based on the registered network provider, which works seamlessly as long as the device does not roam over multiple network providers.

To support network roaming with device manager commands, you can integrate Oracle Lite with Broadbeam ExpressQ (Release 4.1 SP3) and Smart IP (Release 3.1 SP1). Broadbeam enables sending commands across any network type that is supported by

Broadbeam products. Device Manager Commands use the Broadbeam messaging engine. Mobile Client may synchronize data over Smart IP.

7.4 Viewing Device Information

From the Mobile Devices screen, as shown inFigure 7–4, select the device that you want to know more about.

Figure 7–4 Devices Page

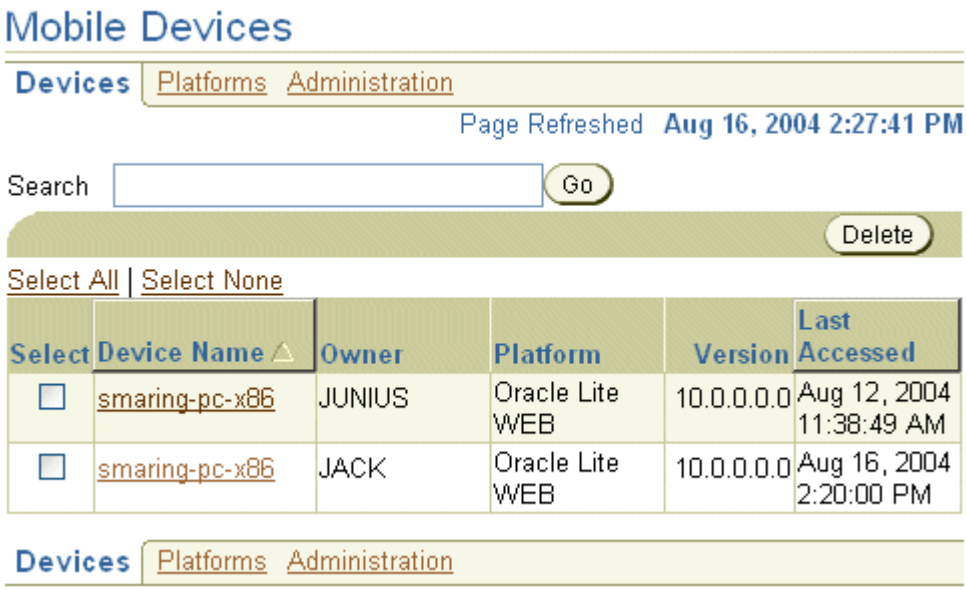
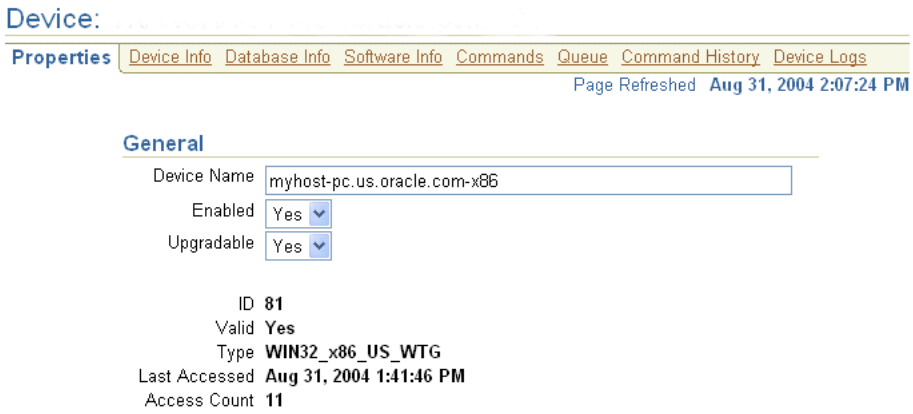


Figure 7–5 shows the specific device page that comes up when you select the device that you are interested in. The Properties screen is the first set of information available.

Figure 7–5 Mobile Device Properties and Information



On the Mobile Device Properties screen, you are told the following about the device:

- Device Name
- Enabled: If the device is enabled. See Section 7.5.2, "Enabling or Disabling All Mobile Devices in a Platform" for more information.

- Upgradable: If the device accepts software upgrades. See Section 7.5.3, "Allowing Software Upgrades to All Mobile Devices in a Platform" for more information.
- Valid: If the Device Manager software is installed and correct. If de-installed, Valid displays No.

Each of the tabs at the top provides more information about the device. Use this information to determine what sort of administration each one needs in order to continue to operate smoothly. You can use this information to tell what needs to be upgraded on each device. The following sections covers each of these tabs.

- Section 7.4.1, "Viewing Device Information"
- Section 7.4.2, "Viewing Database Information"
- Section 7.4.3, "Viewing Software Information"
- Section 7.4.4, "Commands"
- Section 7.4.5, "Queue"
- Section 7.4.6, "Command History"
- Section 7.4.7, "Viewing Device Logs"

7.4.1 Viewing Device Information

The Mobile Manager displays general and database information for a chosen device. To view device information, click **Device Info**. If no information is displayed, click **Retrieve Device Information**. This sends a command to the device, which is then posted back to this page. Click reload until the information is posted.

In the first section, all of the details about the operating system is provided. You no longer have to go to the machine and type in a command to determine the operating system, its version and the latest service pack applied. This section will provide you with all of this information. In addition, you can see what the host name and IP address is.

The second section details how much memory you have on the device. This includes how much virtual or physical memory are on the device, and how much of that memory is still available.

The third section details the type of processor that is installed on the machine. For example, it describes the type of Intel processor that is installed on your Windows machine. You know exactly when your users must be upgraded to the next version of processor for the capability that they need.

For Windows-based devices only, the fourth section details the version of the JDK that you have installed and where it is installed. You no longer have to ask your users to check which version of JDK that they have installed. For example, in the Oracle Database Lite 10*g*, if you are using Oracle Application Server, you must be using the JDK 1.4.2. If you have applied the Oracle Database Lite patch set to the Oracle Database Lite 10*g* release, you can use JDK 5.0. You can view this information for each device and know if the Mobile client software must be upgraded or not. In addition, this section describes the `CLASSPATH` for the Mobile client environment.

The last section details the amount of storage space that exists and is currently available on each drive.

7.4.2 Viewing Database Information

When you select the Database Info tab, you see all of the information about the Oracle Lite database that is installed on the Mobile client.

The first section provides the ODBC driver name, so that you can know which version that is installed on your client. In addition, you can see what DLL is used for the database and the directory.

The second section details each of the user ODB files—that is, the client databases for each application. To validate the file data integrity of the ODB file, select the button next to the ODB file and click **Validate**.

The third section displays the configuration in raw form in the `POLITE.INI` file on the client. Each section in the `POLITE.INI` file is displayed. The purpose of this section is for you to view the sections and parameters in the INI file. The data is shown in raw format.

7.4.3 Viewing Software Information

You can view all of the Oracle Database Lite software that is installed on the Mobile device by clicking **Applications**, which lists each application, its version, setup time, and location details. If no information is available, click **Retrieve Software Information**, which sends a command to the device.

Once the information becomes available, which is dependent on when the device reconnects, the platform is listed for the device. Select the platform to see the software information.

7.4.4 Commands

You can send commands to each device by itself or to all devices in a platform to gather information or execute some function. This is described fully in Section 7.7, "Sending Commands to Your Mobile Devices".

7.4.5 Queue

The queue shows any commands that are currently in process. That is, if the device is not currently connected, then the command is placed into the queue until the device becomes available. Viewing this queue shows you all of the commands that are queued up waiting for devices.

7.4.6 Command History

This shows all of the commands executed against this device.

7.4.7 Viewing Device Logs

The Mobile Manager displays device logs and synchronization logs from the client device. To view the client device logs, click **Device Logs**. The Device Logs page lists what activity has occurred on the device. When you click Purge, these logs are removed.

To view the synchronization logs, perform the following:

1. You must first retrieve the synchronization logs by sending the 'Retrieve synchronization log' command to the device.

2. Then view the retrieved client device synchronization logs by clicking **Synchronization**. This shows only the synchronization requests made by the client.

7.5 Configuring and Customizing Your Mobile Device Platform

Oracle Database Lite ships with a number of predefined Oracle Database Lite Client Platforms that you can download and install on your Mobile client device. However, there are some devices that the CAB file is provided within the installation, but which are not registered and available for download within the Mobile Manager. You can register these devices—if necessary—for any needed device platform. After registration, the platform appears on the Mobile Manager setup screen for your Mobile client installations.

An Oracle Database Lite client platform consists of a CAB file and an Installation Configuration File (INF file) that describes how to install the files.

As described in Section 4.2, "Installing the Mobile Client" in the *Oracle Database Lite Getting Started Guide*, you normally install the Mobile client software by selecting the type of Mobile device and the language in the setup UI. However, you can extend any of these platforms to not only install the Mobile client software for the platform, but also install any of your binaries or applications. The following sections describe how to configure your platform or create and customize a new platform.

- Section 7.5.1, "Modifying Platform Properties for Installation"
- Section 7.5.2, "Enabling or Disabling All Mobile Devices in a Platform"
- Section 7.5.3, "Allowing Software Upgrades to All Mobile Devices in a Platform"
- Section 7.5.4, "Extend or Create a Custom Platform"

7.5.1 Modifying Platform Properties for Installation

Before you install a platform on your system, you should ensure that all of the configuration details for the Mobile device setup are what you want. A setup file is used to detail installation details, such as directories, binaries to install, registry to modify, path and `CLASSPATH` additions, and so on. You can modify the setup INF file that is defaulted for each platform, or you can create your own and point the platform to the new setup INF file.

Generally, you do not want to modify the generic platforms provided for you, in case you need to go back to basics. Thus, you should create your own unique platform by extending one of the provided platforms. This copies the existing platform into a separate platform with your name. Once copied, or extended, modify this platform with your own unique characteristics. For instructions on how to extend one of the provided platforms, see Section 7.5.4, "Extend or Create a Custom Platform".

To modify how the platform is installed, do the following:

1. Designate the name and path of the setup INF file for your platform by navigating to the Mobile Devices page.
   a. Click **Platforms**.
   b. Click on the platform for which you are currently modifying the INF file.
   c. Make sure that the correct setup INF file is listed in the Setup INF field.

Note: If you want to modify this INF file or provide a different INF file, then on the Mobile Server, navigate on the file system to *$OLITE_HOME*/j2ee/mobileserver/applications/mobileserver/setup/dmc to where the setup INF files are located. Open the file that you want to modify or create a new INF file. Add the changes you want for this platform. See Section 7.11, "Installation Configuration (INF) File" for the configuration syntax.

2. Choose a Bootstrap group command from the list displayed. After the device bootstrap is complete, you can choose a group command to execute when it is completed. For example, choosing the device information command retrieves all of the device information to the Mobile Manager for viewing.
3. Enable or disable all devices in the platform. If you enable these devices by choosing Yes, then each user can log in, perform updates, synchronize, and perform other duties. If you disable these devices by selecting No, then they can no longer perform work for the user. See Section 7.5.2, "Enabling or Disabling All Mobile Devices in a Platform" for more information.
4. Set Upgradable to Yes to retrieve all software updates for all devices in the platform. If you want these devices to continue to receive automatic software updates, choose Yes. If you want these devices to stay with the current software versions, choose No. See Section 7.5.3, "Allowing Software Upgrades to All Mobile Devices in a Platform" for more information.

7.5.2 Enabling or Disabling All Mobile Devices in a Platform

You can enable or disable all Mobile devices in a platform. By default, each device in the platform is enabled, which means that the user can synchronize to the database and perform software updates. If you disable the device, then it can no longer perform work for the user. If you wanted to disable a single device—because a user has lost the device or left the company—then you follow the instructions in Section 7.6.1, "Enabling or Disabling a Mobile Device". However, if you want to enable or disable all devices for a platform, then see Section 7.5.1, "Modifying Platform Properties for Installation".

Why would you want to disable all devices in the platform? What if you had created a a customized platform (see Section 7.5.4, "Extend or Create a Custom Platform") for devices that were used for a specific purpose, such as if you had cell phones that were analog only. When you came out with a full digital network, you may not want any of the analog technology to continue to be used. You could choose to send a deinstall command (see Section 7.7, "Sending Commands to Your Mobile Devices") and then disable all of the analog Mobile devices. Since all of them had the same platform, all of them could be disabled at the same time. The user could no longer log in and use the device. They would be forced to upgrade to digital.

7.5.3 Allowing Software Upgrades to All Mobile Devices in a Platform

If you set Upgradable to Yes, then when a new software update comes available—either for the Mobile client software or for any applications installed on the client—then the Mobile device will receive these updates. However, if you want a device to stay with the level of software that is currently installed, then you would change Upgradable to No.

Note: Do not set your device to No until after the first syncrhonization. The device must be configured as upgradable for the first synchronization.

See Section 7.6, "Configuring Your Mobile Devices" for where you set whether the software is to be upgraded or not.

7.5.4 Extend or Create a Custom Platform

You can create custom device platforms either by using an existing platform as the template or by enabling a platform.

- Only a few of the available platforms are displayed in the Mobile client setup screen. To add a platform that you need, enable the desired platform. See Section 7.5.4.1, "Enable a Platform for Your Mobile Client" for more information.
- You can extend an existing platform to customize that platform to install additional binaries, applications, or to have specific instructions on modifying the client machine to accommodate your specifications. See Section 7.5.4.2, "Create a Custom Platform By Extending an Existing Platform" for more information.
- You can create a new platform for an unregistered CAB file. See Section 7.5.4.3, "Create a Custom Platform for Unregistered CAB Files" for more information.

7.5.4.1 Enable a Platform for Your Mobile Client

Not all of the possible platforms are enabled on the Mobile client setup screen. To enable a platform for your client device, do the following:

1. On the Mobile Devices screen, click **Platforms**.
2. On the Platforms screen in the Search pulldowns, select the language and either Disabled or All and click **Go**.
3. Select the platform name that you want to enable.
4. Enable the device by selecting **Yes** in the Enable pulldown.
5. Click **OK**. The device is now enabled and will be visible in the client setup screen.

7.5.4.2 Create a Custom Platform By Extending an Existing Platform

You may wish to install additional binaries, applications, or to have specific instructions on modifying the client machine when the client platform is downloaded to the device. The INF file contains the "directions" to the client on how the platform is installed.

To create a custom platform from an existing platform, do the following:

1. Create a new INF file for your extended platform—On the Mobile Server, navigate on the file system to *$OLITE_HOME*/j2ee/mobileserver/applications/mobileserver/setup/dmc to where the setup INF files are located. Create an empty INF file for your new platform. As discussed in Section 7.11, "Installation Configuration (INF) File", when you extend a platform, the INF files that are used for the installation is a concatenation of your platform and every platform that it was extended from—just like how objects extend methods, properties and attributes from each other in an object-oriented language.
2. On the Mobile Devices screen, click **Platforms**.

3. On the Platforms screen, select the platform name and click **Extend**.
4. Enter the custom platform name, path, and file name of the blank setup INF file you created in step 1. The setup INF file determines what is installed and how the client machine environment is modified. See Section 7.11, "Installation Configuration (INF) File" for instructions on how to modify the setup INF file after you have completed extending the platform.
5. Choose a Bootstrap group command from the list displayed. After the device bootstrap is complete, you can choose a group command to execute when it is completed. For example, choosing the device information command retrieves all of the device information to the Mobile Manager for viewing.
6. Enable or disable the device. If you enable the device by choosing Yes, then the user can log in, perform updates, synchronize, and perform other duties. If you disable the device by selecting No, then it can no longer perform work for the user.
7. Set Upgradable to Yes to retrieve all software updates for the device. If you want the device to continue to receive automatic software updates for the device, choose Yes. If you want the device to stay with the current software versions, choose No.
8. Click **OK**.
9. After you have extended your platform and given it a unique name, you should modify the setup INF file for this platform.

The client can now install your customized platform from the setup UI.

7.5.4.3 Create a Custom Platform for Unregistered CAB Files

Oracle Database Lite ships with a number of predefined Oracle Database Lite Client Platforms that you can download and install on your Mobile client device. However, there are some platforms where the CAB file is provided within the MDK installation, which are not registered, but are available for download within the Mobile Manager. The SDK CAB files are installed with the MDK installation.

You can register these platforms—if necessary. After registration, the platform appears on the Mobile Manager setup screen for your Mobile client installations.

> **Note:** For a quick reference on how to create and register a platform for the WCESTD500 SDK CAB files that contain Java, `msql`, and utilities, see Section 7.5.4.3.2, "Register the WCESTD500 SDK CAB Files with Java, MSQL, and Utility Support".

An Oracle Database Lite 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.

To create your new platform, perform the following:

1. Find the unregistered CAB file for the desired platform and language, as follows:

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

   Copy and rename the CAB file. The CAB files are named `olite.<language>.<platform>.<chipset>_sdk.CAB`. Rename the CAB file to `olite.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.

2. 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. 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 7.11, "Installation Configuration (INF) File".

   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/$CPU$/olite.cab</src>
    <des>$APP_DIR$\olite.cab</des>
  </item>
</file>
```

3. Create an INI file that refers to the INF file for this platform. See Section 7.5.4.3.1, "Defining the INI File" for details.

4. 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
```

7.5.4.3.1 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.
- 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 were discussed in Section 7.5.4.2, "Create a Custom Platform By Extending an Existing Platform" 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
Oracle Lite WCESTD500 ARMV4I;US
#
# Windows CE Standard SDK 5.00 ARM V4i
[PLATFORM.Oracle Lite WCESTD500 ARMV4I;US]
TYPE=WINCE_ARMV4I_US_OLITE_STD500
INF=std500.inf
BOOTSTRAP=DeviceInfo
ATTRIBUTES=dmc=auto&update=true&enabled=true
```

7.5.4.3.2 Register the WCESTD500 SDK CAB Files with Java, MSQL, and Utility Support

The SDK CAB files for the STD500 includes the Java, msql and utility support. These are not automatically included in the setup UI.

The SDK CAB files are installed with the MDK in the following directory:

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

For the following SDK CAB files, these will not show up in the setup UI unless you create a platform for them, as follows:

1. Copy and rename the CAB file. The CAB files are named olite.<language>.<platform>.<chipset>_sdk.CAB. Rename the CAB file to olite.cab. The SDK CAB files are installed with the MDK in the following directory:

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

   After renaming it to olite.cab, copy the SDK CAB file into the appropriate subdirectory according to language and client platform type, as described below:

   Note: Take note of the directory path as you will provide the location of the CAB file in the INF file. in the INF file.

- Oracle Lite WCESTD500 ARMV4I. Copy to the *<ORACLE_HOME>*\mobile_oc4j\j2ee\mobileserver\applications\mobileserver\setup*<language>*\stdsdk500\armv4i directory.
- Oracle Lite WCESTD500 X86: Copy to the *<ORACLE_HOME>*\mobile_oc4j\j2ee\mobileserver\applications\mobileserver\setup*<language>*\stdsdk500\x86 directory.

2. The INF file for these platforms is provided for you in `std500.inf`. If you want to add additional instructions, copy the file and make sure the INI file refers to the new INF file.
3. Create the INI file for this platform.

   For example, the following is the platform-defining INI file for the US version of the Windows CE Standard SDK 5.00 - ARMV4I:

```
 ------------------------------------start---------------------------------
# Platforms
#
[PLATFORM]
# Windows CE Standard SDK 5.00 - ARMV4I
#
Oracle Lite WCESTD500 ARMV4I;US
################### Windows CE Standard SDK 5.00 ############################
[PLATFORM.Oracle Lite WCESTD500 ARMV4I;US]
TYPE=WINCE_ARMV4I_US_OLITE_STD500
INF=std500.inf
BOOTSTRAP=DeviceInfo
ATTRIBUTES=dmc=auto&update=true&enabled=true
------------------------------------ end-------------------------------
```

4. Register the platform with the device manager loader script, as follows:

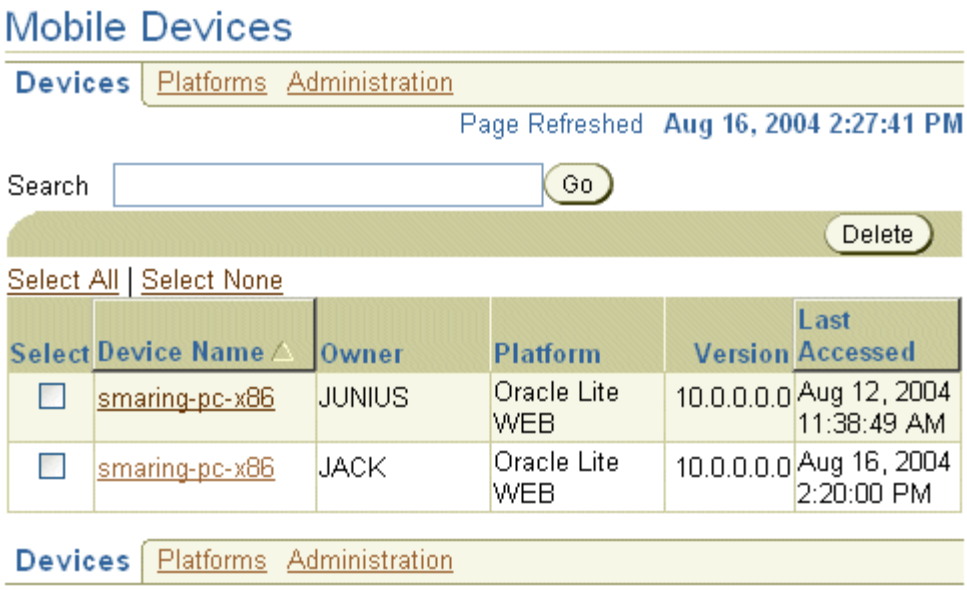
```
ORACLE_HOME\mobile\server\admin\dmloader
        <repository_owner>/<repository_password>@jdbc_url <ini_filename>
```

5. On the client, open a new browser that points to the setup page to select the newly registered platform with the SDK CAB file.

7.6 Configuring Your Mobile Devices

Navigate to the Mobile Devices page, as shown in Figure 7–6, and you can modify, delete, and extend any Mobile device.

Figure 7–6 Devices Page



To modify the configuration of a specific device, select the device and do the following:

1. The device name is displayed on the first line. You can modify the name of the device to anything that you want. For example, if you have a naming convention for all devices in your organization, modify this field to reflect this convention. The name defaults to the Mobile device platform.
2. Enable the device by selecting Yes; disable the device by selecting No. See Section 7.6.1, "Enabling or Disabling a Mobile Device" for more information.
3. Set Upgradable to Yes to retrieve all software updates for the device. If you want the device to continue to receive automatic software updates for the device, choose Yes. If you want the device to stay with the current software versions, choose No. See Section 7.6.2, "Allowing Software Upgrades to the Mobile Device" for more information.
4. The installed address is displayed in the Address field. If the address is an IP address or a phone number, these may change at some point. You can enter the new addresses in this field.
5. Choose the Network Provider type. You can choose a different network provider than that with which you chose to install. The default list includes HTTP, WOR_IAS (Wake on Ring with Oracle Application Server), WOR_SMTP (Wake on Ring with SMTP), SMS, or RAPI. If you added another network provider, these custom network providers will also be included in the list.

Note: To create a network provider, see Section 7.10, "Managing the Network Protocol Between the Device and the Mobile Client Software".

7.6.1 Enabling or Disabling a Mobile Device

You can enable or disable a Mobile device. By default, the device is enabled, which means that the user can log in, perform updates, synchronize, and perform other duties. If you disable the device, then it can no longer perform work for the user. See

Section 7.6, "Configuring Your Mobile Devices" for where you enable or disable the device.

Why would you want to disable the device? If the Mobile device was lost or the user is no longer with the company, but did not return the device, then you might choose to send a deinstall command (see Section 7.7, "Sending Commands to Your Mobile Devices") and then disable the device. This way, the software is no longer on the device and even if the user had another copy of the software to reinstall the application, they could no longer log in and retrieve any information from your company.

7.6.2 Allowing Software Upgrades to the Mobile Device

If you set Upgradable to Yes, then when a new software update comes available—either for the Mobile client software or for any applications installed on the client—then the Mobile device will receive these updates. However, if you want a device to stay with the level of software that is currently installed, then you would change Upgradable to No. See Section 7.6, "Configuring Your Mobile Devices" for where you set whether the software is to be upgraded or not.

7.7 Sending Commands to Your Mobile Devices

As an administrator, you can create and send commands to any Mobile device. These commands can do a range of functions. The following sections describe how to send existing commands to devices and how to create new commands for your own purposes:

- Section 7.7.1, "Scheduling or Sending Commands"
- Section 7.7.2, "Modifying Existing Commands"
- Section 7.7.3, "Creating New Commands"
- Section 7.7.4, "Creating Group Commands"
- Section 7.7.5, "Enabling or Disabling Mobile Device Commands"
- Section 7.7.6, "Viewing the Mobile Device Command History"

7.7.1 Scheduling or Sending Commands

You can send or schedule a command to be sent from the Mobile Manager. Navigate to the Mobile Devices page.

7.7.1.1 Sending Commands

You can send commands to devices that are installed and registered with the Mobile Manager. You can send these commands from several places.

- Sending a command to a single device—To send a command to a single device, select the device from the list displayed on the Mobile Devices page. Select **Commands**. Under the Send Commands section, choose the command—as designated by the description—and click **Send Now**. The Mobile Manager seeks your confirmation and then displays a confirmation message.

  If the command requires arguments, then the Mobile Manager displays an argument collection page. For example, the Upload File command requires a file name as an argument. To send the command to the device, click **Yes**.

- Sending a command to all devices of the same platform type—To send a command to all devices of a certain platform, click **Platform** off of the Mobile Devices page. Click the select button next to the desired platform, select the command from the command pull-down list, and click **Send Command**.

  Note: If a WinCE device is not physically connected to the Mobile Server, then the device manager commands are not sent immediately. Instead, all commands are queued up. The client device receives these commands when connected to the Mobile Server and polls the command queue.

  The default frequency to pull commands is 1800 seconds, which can be configured through the options section of the Device Manager Agent (`dmagent.exe`) located on the client.

For information on how to create a command, see Section 7.7.3, "Creating New Commands".

7.7.1.1.1 Description of Existing Commands The following are the commands that you can use to control your device:

- Retrieve Device Information—retrieves hardware and software information from the device. The command also retrieves Oracle Lite database file names and their sizes. The retrieved information may be viewed by clicking on 'Device Info' and 'Database Info' pages.
- Install Application—Send this command to force the device to install an application. In order for this command to work, the following conditions must be met:
  - The application must be published correctly to the Mobile Server.
  - The application platform must match the device Oracle Lite Platform.
  - The user must have access to the application.
- Retrieve Software Information—Retrieves information regarding Oracle Database Lite and Oracle Database Lite applications. Retrieved information is displayed on 'Software Info' page.
- Stop Device Manager Client—Sends a stop signal to the device manager client. Once the client is stopped, it will not receive any more commands from the server. User must either start the device management agent explicitly (`dmagent.exe`) or invoke 'Check For Update' program in order to restart the device manager client.
- Retrieve synchronization log—Retrieves the data synchronization log from the client. The retrieved information is displayed in 'Device Log->Synchronization' page.
- Synchronize databases—Synchronize all the databases that are 'synchronizable'. Please note that this command does not retrieve the 'synchronization log'.
- Retrieve a file from the device—Force the device to upload a file. By default, this command will retrieve the file from Mobile Client's home directory. If you want to retrieve an arbitrary file, you must provide the full path name of the file. The retrieved file is stored in the Mobile Server's repository and may be viewed by clicking on the hyper-link on the 'Command History' status column. The physical location of the file in the server is `<ORACLE_HOME>\mobile_`

`oc4j\j2ee\mobileserver\applications\mobileserver\devmgr\<USER_NAME>\<DEVICE_ID>`

- Validate Database—Validates the client database and uploads the results to the Mobile Server. The result may be viewed by visiting 'Command History' page and clicking on the status column.
- Synchronize and Delete Databases—Triggers data synchronization and once the synchronization has been completed, it will delete all the databases except '`CONSROOT`'.
- Modify Configuration—Modify configuration settings in `POLITE.INI` or `ODBC.INI` files.
- Update Device Manager Client—Force the device manager client to update the OTL script files. The common use of this command is to propagate the OTL script files copied to the Mobile Server's script directory.
- De-install Oracle Database Lite—De-install Oracle Database Lite client remotely.
- Reset Password—Reset the client side password to match the new password on the server side. This command DOES NOT change the password. In order to use the command, the Administrator must change the user's password in Mobile Server and later send the command to the device to reset its stored password. Also, the device must be immediately reachable from the Server. See Section 2.3.4, "Reset the Mobile User Password" for details.

7.7.1.2 Scheduling Commands

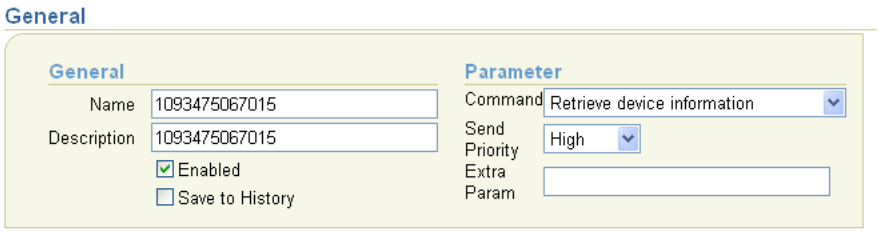
You can schedule a command to execute at a later time or at certain intervals. Select the device to which you want this command to be directed. Select **Commands**. Perform the following:

1. Click **Schedule**. The Schedule Command page appears.
2. As shown in Figure 7–7, configure the following:
   - The name and descriptor are unique identifiers. You can modify them to your own unique identifiers.
   - Choose the command that you want to schedule from the Parameter Command pull-down list.
   - Check the Enabled box to enable or disable the command. If disabled, the command cannot be executed.
   - Check Save to History if you want to keep a log of when this is executed and the results, which are printed to the Command History screen.
   - Choose from the Priority pull-down list if this is to be high, medium, or low priority. This determines in what order the scheduled commands are executed.
   - Enter any expected parameter values, separated by semi-colons, in the Extra Parameter field. For example, if you chose the Synchronize databases command and you wanted a fast refresh, you would enter 'fast' in the Extra parameter field.

Note: If you use complete refresh, it erases all of the data on the client and brings down the snapshot from the server. This is only a problem if you have not specified the publication as updateable. An updateable publication enables all new data entered in the client to be uploaded to the back-end Oracle server.

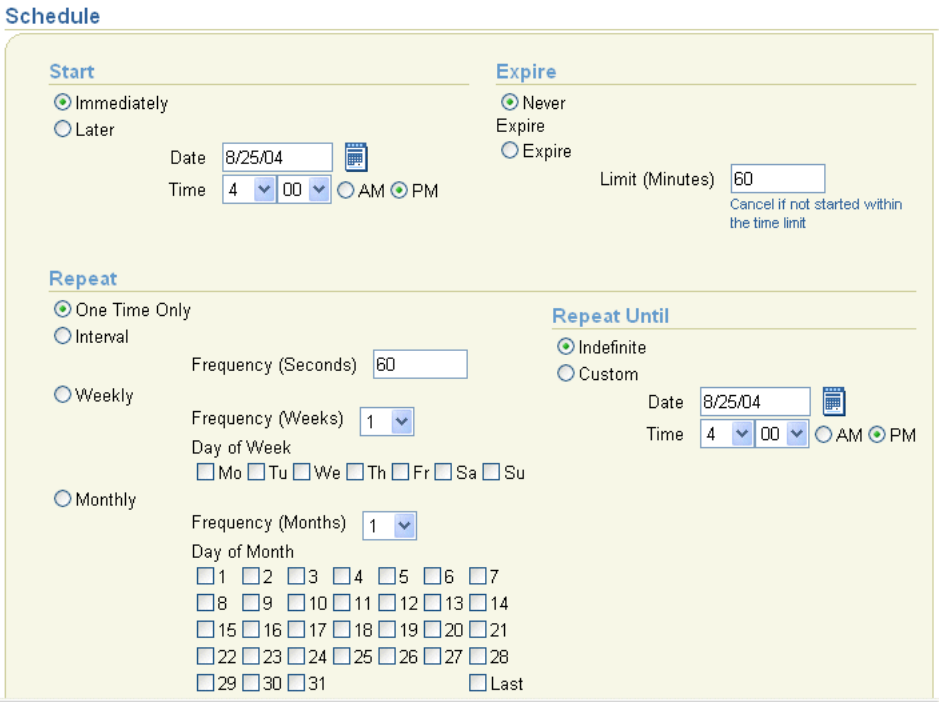
If you use fast refresh, you cannot use it with a high priority restricting predicate.

Figure 7–7 The General Section of the Command Scheduler



3. As Figure 7–8 shows, enter the timing that the command is to execute. Choose when it is to start, when it will expire (if it does not execute within a certain time frame), how often it will repeat, and a date and time that it will repeat until.

Figure 7–8 Timing Section of the Command Scheduler

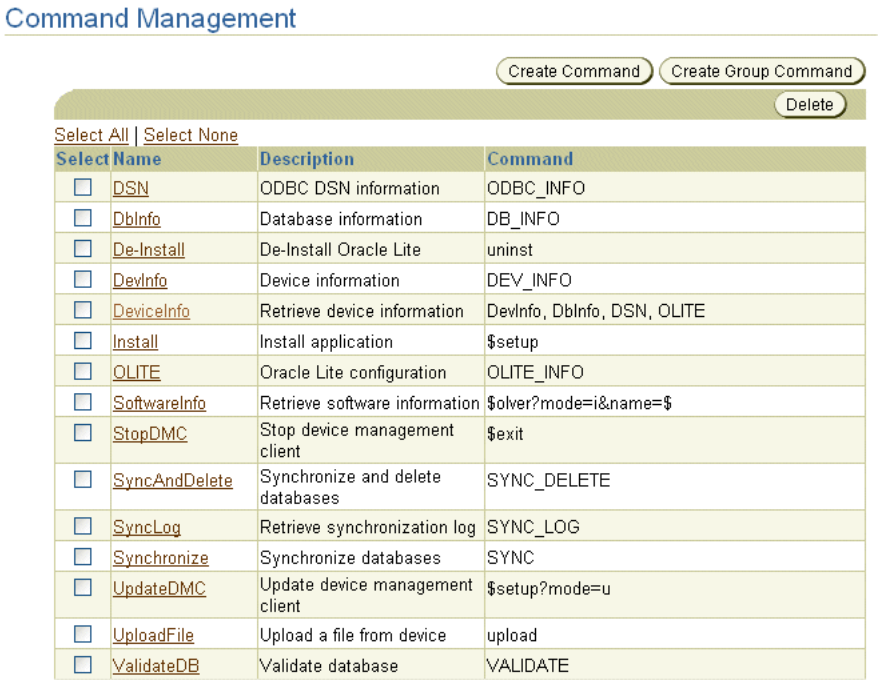


4. Click **Apply**. The Mobile Manager displays a confirmation message.

7.7.2 Modifying Existing Commands

To view the available commands, select the Mobile Devices tab on the Mobile Manager. Navigate to the Administration page and click the Command Management link. As Figure 7–9 displays, the Command Management page appears.

Figure 7–9 Command Management Page



Using the Command Management page, you can modify existing device commands and create new device commands. To modify existing commands, do the following:

1. Click the required Command Name link. The Properties page for this command appears.
2. Enter the command name, description, and syntax in the corresponding fields. For more information on modifying these fields, see Section 7.7.3, "Creating New Commands".
3. To check the accuracy of the command syntax, click **Syntax Check** button. If no errors are found, the Mobile Manager displays a confirmation message.
4. Click **Apply**. The Mobile Manager displays a confirmation message.

7.7.2.1 Adding Parameters to Mobile Device Commands

You must configure the command to prompt for expected input parameters. For example, the Synchronize command requires that you define what type of refresh you want: fast, force, or push.

You can specify any parameters by modifying the command, as follows:

1. Click the required Command Name link. The Properties page for this command appears.
2. Add the parameter name and values, as follows:
   - The parameter name—This is the name specified in the OTL script.

- A short description—The description is what is displayed when the user is prompted for the value of the parameter.
- The display name—This is the description for each value. For example, the Synchronize databases command has three possible values: fast, force or push. However, the display values to describe each of these actual values is Fast Refresh, Force Refresh, and Push Only. These values are separated by a semi-colon, as follows: Fast Refresh;Force Refresh;Push Only.

Note: If you use complete refresh, it erases all of the data on the client and brings down the snapshot from the server. This is only a problem if you have not specified the publication as updateable. An updateable publication enables all new data entered in the client to be uploaded to the back-end Oracle server.

If you use fast refresh, you cannot use it with a high priority restricting predicate.

- The default values for this parameter—Enter one or more potential values for this parameter, if applicable. For example, the Synchronize databases command values would be fast;force;push. These values are separated by semi-colons and in the same order as the display name. If you do not have definitive values, leave blank and the user will enter their own value.

7.7.3 Creating New Commands

You can create commands using the OTL scripting language, as described in Section 7.12, "Defining Device Manager Commands With the Device Manager OTL Tag Language". These commands are then used to perform activity on the Mobile devices, but controlled by the administrator within the Mobile Manager.

You can create a command with a single or multiple OTL script commands. Each is created in a different manner, as described in the following sections:

To create new commands, click **Create Command**. Enter a unique Command ID, Command String, and Description in the corresponding fields. Click the Create button. The Mobile Manager displays a confirmation message.

To create a command that has several lines, you must perform the following:

1. Create a file with an `.otl` extension with the OTL commands in it. Place this file in the *`ORACLE_HOME`*`\mobile_oc4j\j2ee\mobileserver\applications\mobileserver\setup\dmc\otl` directory.
2. With any editor, add all OTL commands that you want executed within the file. See Section 7.12, "Defining Device Manager Commands With the Device Manager OTL Tag Language" for a full description of the OTL scripting language.
3. Within Mobile Manager, navigate to the Command Management page.
4. Click **Create Command**.
5. In the Name field, pick a short name to identify the command within Mobile Manager.
6. In the Command field, put the name of the OTL file, without the `.otl` extension.
7. In the Description field, type in a sentence describing accurately the purpose of the command.

8. Click **OK**.
9. Back on the Command Management screen, select the command that you just created.
10. Click **Syntax Check** to ensure that you entered in your OTL commands correctly.
11. Click **Apply**.
12. If you ask the user to enter parameters, then add the parameter definitions. See Section 7.7.2.1, "Adding Parameters to Mobile Device Commands" for a full description.
13. Push the OTL script to device.

You have now successfully created a new command. You can now execute this command against your Mobile devices. See Section 7.7.1, "Scheduling or Sending Commands" for information on how to execute the command.

7.7.4 Creating Group Commands

To create group commands, do the following:

1. Click **Create Group**. The Create Group Command page appears.
2. Enter a unique Command Name, which will be used to identify the grouping.
3. Enter a description.
4. Select the set of existing commands that you want to execute together. The Command Weight feature controls the order in which the commands are executed. For example, a command with Weight 1 is executed first and a command with Weight 2 is executed next. Users must specify a weight for all the commands for the chosen group command.

> **Note:** If you provide similar weights to more than one command, the commands with the same weight are executed in the sequence in which they are listed on the GUI, which is alphabetical.

5. Click **Add**. The Mobile Manager displays a confirmation message.

7.7.5 Enabling or Disabling Mobile Device Commands

By default, all commands are enabled, which means that you can execute the command. If you want to disable a command, so that it can no longer be executed, do the following:

1. From the Mobile Devices page, click **Administration**.
2. Choose **Command Management**.
3. Select the command that you want to enable or disable.
4. Select either Yes for enable or No for disable on the Enabled pull-down.
5. Click **Apply**.

7.7.6 Viewing the Mobile Device Command History

To view the Device Command History, click the Command History link from the single Mobile device screen. The Command History page lists a history of commands that were implemented for the chosen device. You can delete a single historical

message by clicking the select box next to the message and clicking **Delete**. To delete all messages, click **Purge**.

7.7.7 Examples of Mobile Commands

The following are examples of commands that you can send to your Mobile device:

- To create a command that configures for automatic encryption of a local Oracle Lite database, see Section 5.7.2, "Configuring on the Server for Automatic Encryption of Local Snapshots".
- To trigger a synchronization on the client, see Section 4.3, "Configuring for Default Sync When Installing the Client" in the *Oracle Database Lite Getting Started Guide*.

7.8 Enabling Device Software Updates

This section describes how to enable software updates and patches for your devices.

- Section 7.8.1, "Enabling Major Software Updates for Your Device"
- Section 7.8.2, "Applying Patches or Minor Updates"
- Section 7.8.3, "Controlling Device Software Updates"

7.8.1 Enabling Major Software Updates for Your Device

In order to facilitate a major software update, the corresponding `INF` file must be modified to reflect the new version number. Mobile Server relies on the application version number to determine if the client software is out-of-sync.

To update your software, perform the following:

1. In the software `INF` file, the administrator modifies the version number for the application to the current version, as follows.

   ```
   <setup name="Application Name" version="1.2.3">
   ```

2. The client user synchronizes with Mobile Server. Alternatively, the client user can invoke `update.exe` to check for the latest version of the software.

When the client user synchronizes, Mobile Server compares the client application software version number against the version number in the `INF` file. If the version numbers are different, Mobile Server compares the 'Last Modified Time' of all of the client application files against the server application files to determine the changes and then sends the modified files to the client device.

7.8.2 Applying Patches or Minor Updates

In order to apply specific patches to an existing installation for your application, the application developer creates an `INF` file with the `patch` attribute and copies it to the correct platform patch directory, each of which is located in the *`ORACLE_HOME`*`\j2ee\home\applications\mobileserver\setup\dmc` directory in the Mobile Server.

The following application `INF` file defines the `patch` element as `myFirstApp` for applying a patch to "Application Name" software:

```
<setup name="Application Name" version="1.2.3">
     <property>
        ...
        <patch>myFirstApp</patch>
```

```
    </property>
    ...
</setup>
```

The value in the patch element is a user-defined name. This will be the name of the directory into which the updates for the application are copied. In this example, the updates for "Application Name" are copied into the `myFirstApp` directory.

> **Note:** Be careful to have a unique patch directory name for each application. If you have the same directory name in the `patch` element, then all applications with that patch directory name receive the updates placed there.

In order to update a patch with the `olobj40.dll`, update the patch `INF` file as follows:

```
<setup name="Application Name" version="1.2.3" id='1001'>
   <install>
     <action msg_i='$FILE_I$' msg_u='$FILE_U$'>file</action>
       <file>
         <item>
           <src>/common/win32/olobj40.dll</src>
           <des>$APP_DIR$\bin\olobj40.dll</des>
         </item>
       </file>
   </install>
</setup>
```

> **Note:** For a full description of INF files and the elements within them, see Section 7.11, "Installation Configuration (INF) File".

There are two mandatory attributes in a patch `INF` file, as follows:

- The `INF` file contains a version number, which is the same as the application version number. In the above example, the version number (10.0.0.0.0) tells DMS that the patch is meant for the application version 10.0.0.0.0.
- The `INF` file must have an ID, which is used for determining patch dependencies. This ID can be any number you choose.

If you do have dependencies among patches, then you use the `dependency` element to indicate these dependencies. For example, the previous patch for `olobj40.dll` was configured with the ID of 1001. The following `INF` file configures another patch with ID of 1002. This patch defines a dependency on the patch for `olobj40.dll` by configuring the ID number of 1001 in the `dependency` element.

```
<setup name="Oracle Lite WIN32" version="10.0.0.0.0" id='1002'>
  <property>
    <dependency>1001</dependency>
  </property>
</setup>
```

Update the patch, as follows:

1. The administrator copies the patch `INF` files to the patch directory.
2. The administrator copies the new application files to the application directory.

3. The client user synchronizes with Mobile Server. Alternatively, the client user can invoke `update.exe` to check for the latest version of the software.

The Mobile Server checks for patches and sends all new patches to the client device.

7.8.3 Controlling Device Software Updates

The administrator can control what software updates occur either through modifying the `Upgradable` switch in the Mobile Manager or by programmatically defining what types of updates can occur. There are two types of software update that you can control, as detailed in the following sections:

- Section 7.8.3.1, "Enabling/Disabling Software Updates for the Oracle Database Lite Platform"
- Section 7.8.3.2, "Updating Application Software On Each Client"

7.8.3.1 Enabling/Disabling Software Updates for the Oracle Database Lite Platform

You can enable/disable the Oracle Database Lite platform updates for the devices through one of two methods:

- Setting `Upgradable` to `Yes/No`—Change the value of `Upgradable` in the Mobile Manager GUI to Yes to enable and No to disable, as follows:
  1. Select the Mobile Devices tab in Mobile Manager.
  2. Select the Platform tab to display all Oracle Database Lite Platforms.
  3. Click **Oracle Lite WIN32** platform to display its properties.
  4. Change the value of `Upgradable` to `Yes/No`. Click **OK**.
- Setting the `UPDATE_SOFTWARE` attribute in the Resource Manager to `true/false`. This has to be set programmatically on the Resource Manager object, as follows:

```
rs.setAttribute (ResourceConst.UPDATE_SOFTWARE, "false");
```

If you want to enable/disable any application updates, but continue to allow Oracle Database Lite platform updates, then set the `UPDATE_SOFTWARE_APPS` attribute to `true/false`.

For example, to set the `UPDATE_SOFTWARE_APPS` attribute to false, do the following:

```
rs.setAttribute (ResourceConst.UPDATE_SOFTWARE_APPS, "false");
```

For a full example of how to set the Resource Manager attributes, see *ORACLE_HOME*`\Mobile\Server\samples\devmgr\java\AppUpdate.java`.

7.8.3.2 Updating Application Software On Each Client

You can control whether a new version of an application software is downloaded on each client. Modify the update policy attribute of the user in one of the following ways:

- On the user page in the Mobile Manager, set the Software Update pulldown to the appropriate update that you want, as follows:
  - All updates—Include major and minor updates.
  - Major—The devices attached to this user receives only major software updates. This is the default.

- Minor—The devices attached to this user receives only minor software updates.
- Disable updates—The devices attached to this user does not receive any software updates.

In addition, you can specify the date that the update occurs.

- Set the `UPDATE_SOFTWARE_APPS` policy attribute of the `User` object to one of the following values to specify what type of update that the client can receive:
  - `Major`—The devices attached to this user receives only major software updates. This is the default.
  - `Minor`—The devices attached to this user receives only minor software updates.
  - `False`—The devices attached to this user does not receive any software updates.

```
user.setPolicy (ResourceConst.UPDATE_SOFTWARE_APPS, "Minor");
```

For a full example of how to set the Resource Manager attributes, see *ORACLE_HOME*`\Mobile\Server\samples\devmgr\java\UserPolicy.java`.

Example 7–1 Upgrading Devices Attached to a Specific User

Each user owns one or more devices. You can configure it so certain users do not receive the latest update. The default is that all devices attached to a user receive current updates.

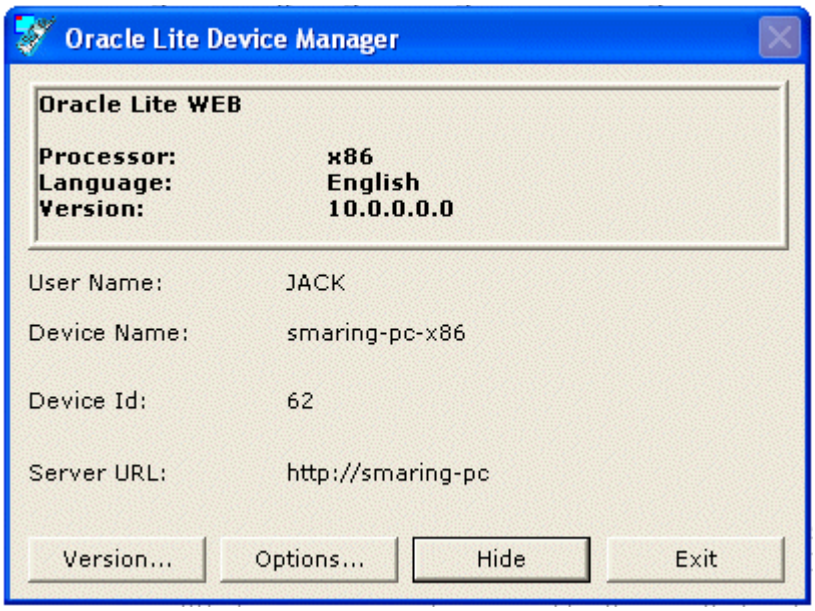
For example, you have two users: John and Tom. You want John's devices to stay at the current version, which is Oracle Database Lite Win32 version 10.0.0.0.0; however, you want Tom's devices to upgrade to the new version, which is Oracle Database Lite Win32 version 10.1.0.0.0. Configure each user's devices, as follows:

- For John, configure the `update.software.apps` attribute to `Minor`.
- For Tom, configure the `update.software.apps` attribute to `Major`.

7.9 Using the Device Manager Agent (dmagent) on the Client

On any client, you can manage the Mobile device client software and commands sent to the device from the Mobile Server, as shown in Figure 7–10.

Figure 7–10 Using the Oracle Database Lite Device Manager to Manage Your Device



To bring up the Device Manager GUI, choose **Oracle Lite Device Manager** from the Oracle Database Lite Programs list or execute `dmagent`. The main screen provides information about the following: platform with type of platform, language, and version, the owner/user of this device, the name of the device, and the URL of where the device is installed.

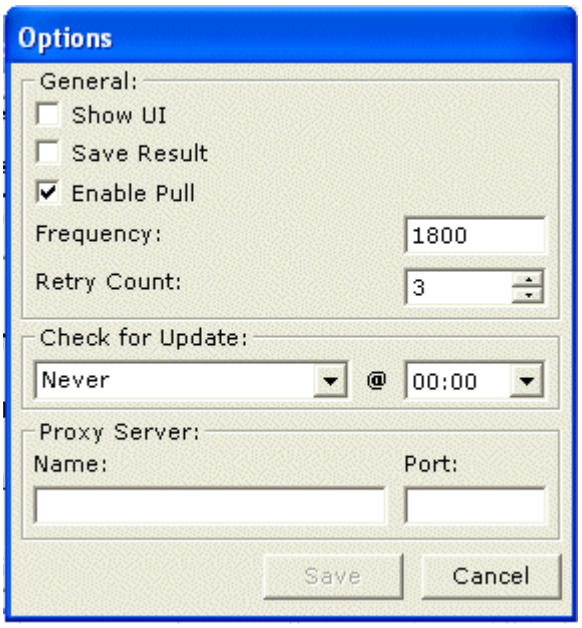
- Click **Version** to see the version number of all Oracle Database Lite software DLLs.
- Click **Options**, which brings up Figure 7–11, to configure the following:
  - Enable Pull: If the server cannot connect to the client, any commands sent to the client are placed into the Device Manager command queue. These commands are sent to the client when one of two things occurs:
    * The client synchronizes.
    * A "Pull" is initiated from the client. When you check the **Enable Pull** checkbox, the device manager client automatically polls the Device command queue for any commands for this user. The frequency is the number of seconds to wait between each pull.
  - Retry Count: If you have created your own Device Command, this count specifies the number of times that the command is executed, if it fails to execute. If it still fails after retrying, the command is deleted. This count also applies to failed synchronization attempts. The commands are retried with the same frequency interval that is set for the Enable Pull command. A command can fail to execute if there is an error within the command or if there is no connection between the client and the server.
  - Check for Update: Select the day and time that you want the device manager client to automatically poll for updates. This also occurs anytime you start a synchronization. So, this is useful if you never synchronize with the Mobile Server.

    This specifies around the time of day to initiate the update tool, which checks for software updates for the client. The actual time depends on when the device manager checks for queued commands. See Section 2.3.6, "Initiate

Updates of Oracle Database Lite Software from the Client" for more details on the update tool.

– Proxy Server: If you have a proxy server between the Mobile client and Mobile Server, enter the address and port here.

Figure 7–11 Device Client Manager Options



- Click **Hide** to place the **Oracle Lite Device Manager** in the Windows System Tray.

7.10 Managing the Network Protocol Between the Device and the Mobile Client Software

The Network Management page is where the administrator defines the properties of an installed network provider or register new network providers. A network provider is the protocol that the Mobile client uses to communicate between itself and the Mobile device. The Mobile client software, which is often installed on a Windows system, sends commands to the Mobile device over this protocol. Often, you have a device, such as WinCE, that interacts with the Mobile client installed on a Windows system.

This network provider definition describes what you have already installed as the protocol between the Mobile client and the device. The frequently-used network providers are as follows:

- HTTP—If you use HTTP, you will provide an HTTP URL in the Address field. You cannot use HTTPS between the Device Manager and the Mobile device.

  Note: HTTP may not work if the device does not have a direct IP conection to the Mobile Server.

- RAPI—Remote API used by the ActiveSync API, which only supports WinCE class devices. These devices connect directly to the computer that is executing the Mobile Server.

  - RAPI does not work on LINUX or UNIX-based systems.
  - When using RAPI on a Windows machine, install and configure ActiveSync 3.7.1.
  - Set `CEUTIL.DLL` and `RAPI.DLL` in the `%WINDIR%\System32` directory.
  - In addition, if your Mobile Server uses Oracle Application Server as its middle-tier solution, set `%WINDIR%\System32` in the application server path.
- Wake on Ring—If you have a cell phone as a Mobile device, then you would have a network protocol where the cell phone receives incoming data. Thus, the address is a phone number for the cell phone. The cell phone is "woken" when incoming commands are initiated from the Mobile Manager. There are two types of Wake on Ring:
  - WOR_SMTP: Wake on Ring over SMTP.
  - WOR_IAS: Wake on Ring over Oracle Application Server.
- SMS—Short Messaging Service.

To modify or create a new network provider, navigate to the Devices page, click **Administration**, and click **Network Management**. The Network Management page lists existing network providers. Select any of these providers to see their properties, which consists of the following:

- Java classname: The name of the Java class that implements the Network protocol, such as HTTP, SMS, EMAIL, and so on.
- Metadata: Any user defined string that is required as input by the Java class during the initialization. See Section 11.5.1.5.3, "Proxy Configuration for the Mobile Server" for an example of how to configure the metadata for the HTTP protocol.

To define a new network protocol, do the following:

1. Create a Network Provider class using Java. This class must implement the `oracle.lite.provider.NetworkProvider` interface.
2. Register the Network Provider through the Mobile Manager on the Network Management page, as follows:
   a. Click **Create** on the Network Management page.
   b. Input the network provider name, Java class name, and metadata.
   c. Click **OK**.

7.11 Installation Configuration (INF) File

The Installation Configuration file contains all the instructions required to install or de-install client software and its format is based on XML. The INF file contains a set of actions and each action may have multiple items.

When you extend a platform, the INF files that are used for the installation is a concatenation of your platform and every platform that it was extended from—just like how objects extend methods, properties and attributes from each other in an object-oriented language. For example, the Branch Office INF file is extended from the Web-to-Go INF file. Thus, when you install Branch Office, the Mobile Server concatenates instructions for the installation from both the Branch Office and Web-to-Go INF files. Any modifications added to the Web-to-Go INF file since the extension will still apply to the installation as both INF files are read at installation

time. When you view the configuration information on Mobile Manager, the type field describes all of the platforms that this INF file extends from.

As Table 7–2 describes, the supported keywords start with a '$' character and ends with a '$' symbol.

Table 7–2 Software Management Client Keyword Description

Keyword	Description
APP_DIR	Application directory of the application
APP_NAME	Application name
OS_DIR	Operating system directory
OS_TYPE	Operating system type, which can be one of the following: WIN32, WINCE, LINUX.
OS_VER	Operating system version. For example, NT, 95, XP, 3.0, and so on.
OS_LANG	Language or Location name, which can be US for English or JA for Japanese.
$DESKTOP$	Folder name of the Windows desktop.
CPU	Device processor type. For example, x86, ARM, MIPS, and so on.
$HOST_NAME$	Host name of the client device.
$USER_NAME$	User name
$HTTP_PROXY$	HTTP Proxy Server URL, if any.
$SERVER_URL$	Oracle Mobile Server URL.

The following sections describe the INF file:

- Section 7.11.1, "Setup Information"
- Section 7.11.2, "Properties"
- Section 7.11.3, "Initialization"
- Section 7.11.4, "Including Other INF Files"
- Section 7.11.5, "INSTALL Element"

7.11.1 Setup Information

All Software Management actions are enclosed within the SETUP XML tag. The SETUP consists of a set of PROPERTIES, INITIALIZATION, INCLUSION of other INF files and INSTALLATION actions. All the four items must be child elements of the SETUP element. A sample INF file is given below.

```
<setup name="Oracle Lite" version="1.0.0.0">
<property>…. </property>
        <init>…</init>
        <include>…</include>
        <install> …..</install>
</setup>
```

Setup may have the following attributes specified as XML tag attributes.

1. NAME - Application name (Mandatory).

2. VERSION - Application version number (Mandatory).
3. PACKAGE - Package Type, which can only be `cab` to specify a Windows CAB format.

7.11.2 Properties

All of the `SETUP` properties must be the child element of the `PROPERTY` tag. Setup may have following properties.

- `STORAGE` - Estimated disk (storage) space (in MB) required for an application.

  ```
  <storage>5</storage>
  ```

- `MEMORY` - Minimum amount of system memory in MB. Required.

  ```
  <memory>5</memory>
  ```

- `LOCATION` - Location or directory name of the application. You can specify the location for the application in one of the following ways:
  - Default directory. If you do not specify anything or specify `<location></location>`, then the directory defaults to `mobileclient`, which is created on the drive with the largest available free space. The user is always prompted to either accept the default or enter the directory that they wish.
  - Absolute directory. Define the absolute path where the application is to be installed. The following installs the application in the `c:\abc` directory:

    ```
    <location>c:\abc</location>
    ```

  - Specify a default directory name. The directory will be created on the drive with the largest available free space. In addition, the user is always prompted to be able to alter the directory into which the application is installed. The following example defines the default directory as `abc`:

    ```
    <location default='abc'></location>
    ```

  - Define a default directory and an absolute directory. You can specify an absolute directory, where the drive may not exist. If the drive does not exist, then a prompt appears with the default directory, where the user can accept he default or provide another. The following defines the absolute directory of `e:\abc`, which if the E drive does not exist, then the default directory of `abc` is created on the drive with the most available free space:

    ```
    <location default='abc'>e:\abc</location>
    ```

  - Specify the platform(s) that this application is installed upon. You can define, with the type attribute, what platforms this application is to be installed on or not installed on. The platforms that you can specify are WIN32, LINUX, and WINCE.
    * To install only on WIN32, do the following:

      ```
      <location default='abc' type='WIN32'></location>
      ```

    * To install on all platforms, except WIN32, do the following:

      ```
      <location default='abc' type=!'WIN32'></location>
      ```

    * To install on either WIN32 or WINCE, do the following:

      ```
      <location default='abc' type='WIN32|WINCE'></location>
      ```

- PROMPT - You can have a window pop-up with a prompt if one of the files that you need to install is currently being used. If the user inputs Yes, then the other instances using that file are terminated. For example, if other executables are using the ologj40.dll when you are installing the client, the prompt "Would you like to terminate the Oracle Lite Application?" is provided to the user.

  <prompt><item type='WINCE' file='olobj40.dll'/>*Would you like to terminate the Oracle Lite Application?*</prompt>

  The following is an example of the setup section in the INF file:

```
<setup name="Oracle Lite" version="1.0.0.0">
<property>
        <storage>4</storage>
        <memory>12</memory>
        <location>d:\tmp\a</location>
        <prompt>
            <item'>Would you like to install App1?</item>
            <item file='olobj40.dll'>Would you like to close
                      Oracle Lite Applications?></item>
        </prompt>
</property>
```

7.11.3 Initialization

Initialization includes setting keywords that you can use when installing your application; the Oracle Database Lite installation keywords are described in Table 7–2. Specify a keyword for your application installation in the type parameter and its value in the name parameter. The following defines a WIN32 keyword with a value of APP_DIR/bin.

```
<init> <item type='WIN32' name='DMC_DIR'>$APP_DIR$/bin</item> </init>
```

7.11.4 Including Other INF Files

The following syntax allows an INF file to include other INF files:

```
<include>/dmc/common/webtogo.inf</include>
```

The value of this tag can be an application name or a fully qualified INF file name. If the value is an application name, the DMS includes the INF file of the application.

7.11.5 INSTALL Element

This section lists all the installation steps necessary to perform Software Installation. Each of the steps (actions) must correspond to another child entry or tag. Each action element has a set of ITEMS and two optional caption strings. The caption string is displayed on the SMC user interface. For example,

```
<action msg_i='Creating directories' msg_u='Removing
directories'>directory</action>
```

When the SMC interprets the above tag, it looks for a child element by the name directory and processes all the child items of this element. At this stage, the Device Manager UI indicates that directories are being created. If you have a child element without a corresponding action element, it will not be executed. The action elements force the invocation of the child elements.

Table 7–3 describes INSTALL actions that are supported by the SMC.

Table 7–3 INSTALL Actions Supported by the SMC

Action	Description
directory	Lists all directories to be created.
file	Lists all the files to be copied.
env	Lists all the environment variables to be added to the Operating System.
registry	Registry keys and values to be added to the Windows Registry.
odbc	ODBC driver and DSN to be created.
java	JRE to be installed in the computer.
link	Folder links to be created. For example, desktop, menu, and so on.
ini	INI (configuration files) to be updated.
register	DLLs to be registered with Windows.
execute	Executable files to be launched during the installation process.
finish	Installation completion messages.

DIRECTORY

The directory element contains names of all the directories to be created during the installation process. Entries in this section are fully qualified directory names. For example,

```
<directory>
    <item>$APP_DIR$\oldb40</item>
    <item>$APP_DIR$\crm</item>
</directory>
```

The SMC creates OLDB40 and CRM directories in the ROOT of the application directory.

FILE

The file element lists all the files to be copied during the software installation process. Each item contains a target file name and source file name. The source file name must be unique. We do not support copying the same source file to multiple destination files.

```
<file>
   <item>
        <src> win32/crm/crm.dll </src>
        <des>$APP_DIR$\crm\crm.dll </des>
   </item>
</file>
```

If you want to copy a source file multiple times, you cannot just define the source file and then configure for it to be copied to multiple destinations. Instead, you must manually copy the source file to another filename and then configure it as follows:

```
<file>
  <item>
        <src> win32/crm/crm.dll </src>
        <des>$APP_DIR$\crm\crm1.dll </des>
   </item>
   <item>
        <src> win32/crm/crm2.dll </src>
```

```
        <des>$APP_DIR$\crm\crm2.dll </des>
   </item>
</file>
```

Where `crm.dll` and `crm2.dll` are the same source file.

The file element also supports inflation of JAR and ZIP files. To inflate a file, use the `inflate='true'` attribute along with the item tag. In the following example, the `inflate` tag copies the `client.jar`. Once copied, the `abc.jar` file is inflated into the `$APP_DIR$/bin` directory.

```
<file>
    <item inflate='true'>
        <src>/common/win32/client.jar</src>
        <des>$APP_DIR$\bin\abc.jar</des>
    </item>
</file>
```

ENV

The `ENV` element contains all environment variables to be added to a Windows NT registry. This modifies only the User environment in Windows NT systems.

```
<env>
   <item name='PATH'>$APP_DIR$\WEBTOGO</item>
</env >
```

The above example appends the `application_root\webtogo` directory to the `PATH` environment variable.

REGISTRY

The `registry` element modifies or removes Windows Registry values. All the entries in this section must be a fully qualified registry key name. Sub key names and values must be specified as a sub section. For example,

```
        <registry>
            <item>
                <key>HKEY_CURRENT_USER\Software\Oracle\Test</key>
                <item name="Count" type="DWORD">400</item>
                <item name="Test" type="STRING">ABCDE</item>
            </item>
        </registry>
```

The SMC adds the Windows Registry key named `Test` in the directory named `HKEY_CURRENT_USER\Software\Oracle` and creates a String value named `Test` and a DWORD value named `Count` inside the key. If the same script is used in `UNINSTALL` mode, the SMC removes the key from the Registry.

ODBC

This section creates the ODBC driver and DSNs in the client device. For example,

```
 <odbc>
  <item name="driver:Oracle Lite 40 ODBC Driver" dll='$APP_DIR$\bin\olod2040.dll'>
   <version>02.00</version>
   <admin>$APP_DIR$\bin\olad2040.dll</admin>
  </item>
  <item name="driver:Oracle Lite 40 ODBC Driver (Client)" dll='$APP_
DIR$\bin\olcl2040.dll'>
   <version>02.00</version>
   <admin>$APP_DIR$\bin\olclad2040.dll</admin>
```

```
  </item>
  <item name="dsn:POLITE" driver='Oracle Lite 40 ODBC Driver' dll='$APP_
DIR$\bin\olod2040.dll'>
   <Data_Directory>$APP_DIR$\OLDB40</Data_Directory>
  </item>
 </odbc>
```

JAVA

The `JAVA` element lists the JRE file name and the expected JAVA version. If the expected JAVA version is greater than the version that is already existing in the computer, the SMC installs a new JRE, which is downloaded from the Mobile Server.

```
<java version="1.3.1">
    <item>
        <jre>webtogo\j2re-1_3_1_01-win.exe</jre>
        <iss>webtogo\jre_setup.iss</iss>
    </item>
</java>
```

LINK

The `LINK` element creates a symbolic link on the client system, such as a UNIX soft link or a Windows program link (or Program menu item). Each entry must have a name, a program file name and a folder name, which describe where you want to put the symbolic link and the file path.

The following example creates a symbolic link on a UNIX platform to `libolite40.so.1` in `$ADD_DIR/bin` directory, which points to `$APP_DIR$/bin/libolite40.so`.

```
<link>
    <item name='libolite40.so.1'>
        <folder>$ADD_DIR/bin</folder>
        <file>$APP_DIR$/bin/libolite40.so/file>
    </item>
</link>
```

For Windows platforms, you can also optionally set the current working directory with the `<directory>` tag and the default arguments can be set using the `<arg>` tag. The following example creates a Windows program link to `Oracle Web-to-Go.lnk`, where the `.lnk` is automatically appended, in the `Startup` folder for the `webtogo.exe` file, which is located in the `$APP_DIR\bin` directory.

```
<link>
    <item name='Oracle Web-to-Go'>
        <folder>Startup</folder>
        <file>$APP_DIR$\bin\webtogo.exe</file>
        <directory>$APP_DIR\bin</directory>
    </item>
</link>
```

INI

This section creates entries in the INI (configuration) files. Each item must have an INI file name and a set of values to be added to a section. For example,

```
<ini>
    <item name="POLITE.INI" section="All Databases">
            <item name="DATABASE_ID">200</item>
            <item name="NLS_LANGUAGE">ENGLISH</item>
    </item>
```

```
</ini>
```

If you want to replace an existing item in the INI file, just provide the name and value, such as follows:

```
<item name="Data_Directory">c:\mobileclient</item>
```

The replace attribute defaults to true; thus `Data_Directory` is modified—whether it already exists or not in the INI file—to be `c:\mobileclient`. However, if you do not want to overwrite any existing value, but want only to add `Data_Directory` if it does not exist, then set the `replace` attribute to false. The following example only adds `Data_Directory` with `c:\mobileclient` if `Data_Directory` is not currently configured in the INI file. If it is configured, the value is not replaced.

```
<item name="Data_Directory' replace="false">c:\mobileclient</item>
```

The default value for the `replace` attribute is true; thus, if you want to replace the value, then set the `replace` attribute to false.

EXECUTE

The `EXECUTE` element lists all the programs to be executed during the installation process. Each item must have a program name, wait period, and program arguments. The wait value defines how long the installer waits until it moves on to the next action. The wait value can be either an event name, multiple event names, or time specified in seconds. If the value is seconds, then the installer waits that many seconds and then moves on to the next action. However, if the wait value is an event name(s), then the installer waits for the executable (in this case, the `webtogo.exe`) to post that event, before the installer moves on to the next action. For example,

```
<execute>
   <item>
    <file>$APP_DIR$\webtogo\webtogo.exe</file>
    <args>-h</args>
    <wait>WebToGoSetupExit/WebToGoSetupStop</wait>
   </item>
</execute>
```

REGISTER

The `REGISTER` element lists all DLLs to be registered with the Windows Operating System. For example,

```
<register >
    <item>$APP_DIR$\webtogo\msync_com.dll</item>
</register>
```

7.12 Defining Device Manager Commands With the Device Manager OTL Tag Language

You can send a command from the Mobile Server to any device. To create these commands, use the Device Manager Tag Language, which is described in the following sections:

- Section 7.12.1, "Device Manager Tag Language Data Types"
- Section 7.12.2, "Operators That You Can Use With the Device Manager Tag Language"
- Section 7.12.3, "Syntax for the Device Manager Tag Language"

7.12.1 Device Manager Tag Language Data Types

The allowed data types for the device manager are as follows:

7.12.1.1 Character

The Character object represents a UNICODE character. It is a primitive data type with no public methods. However, this data type supports implicit conversion methods, such as `toString()`.

7.12.1.2 Number

The Number data type represents either an integer, a double (float), or a large number.

7.12.1.3 Integer

The Integer object represents a four byte signed value. It is a primitive data type with no public methods.

7.12.1.4 Long

The Long object represents an eight byte signed value. It is a primitive data type with no public methods.

7.12.1.5 Double

The Double object represents a signed double (float) value. It is a primitive data type with no public methods.

7.12.1.6 Boolean

The Boolean object has only two possible values of `true` or `false`. It is a primitive data type with no public methods.

7.12.1.7 String

The String object represents a series of NULL terminated characters. The String data type represents all of the literal strings in OTL. They are immutable and has the following public methods:

Length ( )

Returns the number of characters in the string.

SubString (Integer start, Integer end)

Creates a sub-string from a String object. Provide the start and the end of the index and it returns the sub-string—beginning at the start value and stopping at the end value.

Trim ( )

Trim a string to remove white spaces from both ends of the string.

IndexOf (Character ch) or IndexOf (String str)

Find the index of a character or a substring within the string. Provide the character or substring inside the string to search for and the index of the first occurrence of the character or substring is returned.

LastIndexOf (Character ch) or LastIndexOf (String str)

Find the index of the last occurrence of a character or a substring within the string. Provide the character or substring inside the string to search for and the index of the last occurrence of the character or substring is returned.

EqualsIgnoreCase (String str)

Compares two strings, without comparing the case of the characters within the string. On one string, execute this method and provide the string to compare it to within the input parameter. True or false is returned.

StartsWith (String str)

Check if the string starts with the provided sub-string. True or false is returned.

EndsWith (String str)

Check if the string ends with the provided sub-string. True or false is returned.

ParseNumber ( )

Parse the string and create a number. This method succeeds only if the string represents a valid number. A number object is either an Integer, Double, or Long. For example, if the content of the String is '12', then this method returns an Integer of 12.

Replace (String in, String repl)

Replace a substring with another, as follows:

```
<c:set var='str' value='${str.Replace ("123,"345")}'/>
```

ToUpperCase ( )

Converts characters in the string to upper case.

ToLowerCase ( )

Converts characters in the string to lower case.

Tokenize (Character sep)

Tokenize the string into sub-strings, each separated by a character separator. The input parameter is the character that is to be used as the character separator. The output is an Enumeration object. For example, the following OTL script separates numbers by separating a string everytime it encounters a semi-colon:

```
<c:set var="str" value="1;2;3;4"/>
<c:foreach var="tok" items="${str.Tokenize (';')}">
 Token = <c:out value="${r}"/>
</c:foreach>
```

7.12.1.8 Array

The OTL Array object can hold a set of other objects. An array can hold dissimilar objects and can grow automatically as more objects are added. All of the array objects have a global scope.

Sort (Boolean ascend)

Sort the content of the array using a Quick Sort algorithm. The array must be single dimensional. Returns the Sort order.

Length ( )

Returns the size of the array.

Compact ( )

Removes all of the NULL objects from the array.

Copy (Integer from, Integer count)

Copy a number of elements within the array to another array. Give the place in the index to start the copy in the from parameter and the number of elements to copy in the count parameter. An array containing these copied elements is returned.

Insert (Integer index, Object o)

Insert the element provided in `Object` o into the spot designated by the index parameter.

Remove (Integer index)

Remove the element in the array at the location of the index parameter.

7.12.1.9 Date Methods

Use `System.Date` to create a `Date` object, which contains the date and time. The following are other methods that pertain to dates.

GetYear ( )

Retrieve the year out of the `Date` object.

GetMonth ( )

Retrieve the month represented by an integer from 1 to 12 out of the `Date` object.

GetDay ( )

Retrieve the day of the week where Sunday is 0 to Saturday, which is 6, out of the `Date` object.

Format (String format)

Format the date as described by the format string, which can be either `dd/mm/yyyy` or `mm/dd/yyyy`.

IsLeapYear ( )

Check if the year of the date is a leap year or not. Returns true if it is a leap year; false if not.

7.12.1.10 Time Methods

A `Time` object represents `Time` value. A `Date` object always contains a `Time` object. You can also create a `Time` object using the `System.Time` function. In addition, the following methods pertain to time:

GetHour ( )

Retrieve the hour out of the `Time` object.

GetMinute ( )

Retrieve the minute out of the `Time` object.

GetSecond ( )

Retrieve the second out of the `Time` object.

Format (String format)

Format the `Time` object using the provided format string. The format string should either be `hh:mm:ss` or `hh:mm`.

To12Hour( )

Convert the `Time` object to a 12 hour format instead of a 24 hour format.

7.12.1.11 Enumeration

Contains a list of objects. Some of the object types that can be contained in the Enumeration object is a SQL result set, a String Tokenizer, or Request Parameter names.

Count ( )

Counts the number of elements in the Enumeration object. Returns the number of elements.

Next ( )

Accesses the next element in the Enumeration object.

7.12.1.12 File

An object of this type can be used to access contents of a file in the file system. You must use the `OpenFile` function to open an existing file (a `System` function). OTL does not allow creation of new files or modification of existing files.

Exists ( )

True is returned if the file exists in the file system.

Open ( )

Open the file for reading. Throws an exception if the file does not exist.

ReadLine ( )

Returns a string from the open file. It reads a line from the file that is terminated by a `\r\n`.

7.12.2 Operators That You Can Use With the Device Manager Tag Language

You can use operators for calculations on certain objects, as

Table 7–4 Device Manager Tag Language Operators

Operator	Description
+	Use can add numbers within Integer, Long, or Double objects. If applied to a String, the strings are concatenated.
-	Subtract numbers contained in Integer, Long, or Double objects. Subtract dates or time.
*	Multiply numbers contained in Integer, Long, or Double objects.
/	Divide numbers contained in Integer, Long, Double, or Character objects.
%	A mod operator applied against Integer, Long, Double, or Character objects.

7.12.3 Syntax for the Device Manager Tag Language

OTL supports all regular scripting language syntax rules, such as Assignment, Conditional Constructs, and Sub Routines.

7.12.3.1 Initialization Statements

You can define primitive data types or arrays using the following:

- Defining primitive data types: Use the `SET` syntax to define a new variable in OTL. `SET` can also be used as an Assignment statement.

```
<c:set var="a" value="1"/>
<c:set var="b" value="String variable"/>
<c:set var="c" value="${a}"/>
```

- Defining an array: Use the `SET` syntax to define the array, which can hold any type of object, including mixed types.

```
<c:set var="arr1" value="${{"aa", "bb", "cc"}}"/>
<c:set var="arr2" incex="0" value="10"/>
<c:set var="arr3" value="${{}}"/>
```

  The first array, `arr1`, is initialized with the values provided. The second array, `arr2`, is initialized with the number 10 at index 0. The third array, `arr3`, creates an empty array. When values are assigned to an existing array, the array is expanded, as necessary.

  The following example expands the array, `arr1`, to size 11. All of the values from index 3 to 9 is set to NULL.

```
<c:set value="${arr1.insert(10, "dd")}"/>
```

  If there already was an object at location 10, then the object is replaced with the new object, "dd". To insert a new object at index 10 and keep existing data, use the `Insert` method, as follows:

```
<c:set value="${arr1.insert(10, "dd")}"/>
```

7.12.3.2 Assignment Statements

`SET` and `SQL` are two distinct assignment syntax statements.

- `SET` supports normal operations, such as arithmetic operations. Normal arithmetic operations can be used on most of the primitive data types, as well as other

objects. OTL converts data types appropriately when arithmetic operations are applied to objects.

- SQL executes SQL statements on a database connection, which results in a SQLRESULTSET object.

```
<c:set var="a" value="1"/>
<c:set var="a" value="${a + 2}"/>
<c:set var="b" value="${"1" + 2}"/>
<c:set var="dt" value="${System.Date ("01-01-2004")}"/>
<c:set var="dt" value="${dt + 1}"/>
```

In this example, a is first assigned the value of 1. Then, two is added to a, which brings the value to three. The value of b is initialized to 12 and dt is initialized to the date of Jan. 1, 2004. Lastly, a 1 is added to dt, bringing the date value to 01-01-2004.

7.12.3.2.1 Creating a SQL Result Set Use the SQL syntax to create a SQL Result. SQL syntax is similar to the SET syntax. The following example assigns a SQL statement to the rs variable.

```
<c:sql var="rs" value="select table_name from all_tables"/>
```

7.12.3.2.2 Print Value to the Output Stream Use the OUT syntax to print a value to the output stream object, as follows:

```
<c:out value="${a}"/>
```

7.12.4 Conditional Statements

OTL supports the following four types of conditional statements:

- If-Else
- While
- Foreach
- Choose

Each statement must end with the appropriate end tags. Conditional operators, such as &&, ||, ==, >, >=, <, <=, and != are supported by OTL. However, implicit boolean conditions are not allowed, such as if (value).

7.12.4.1 If-Else Conditional Statement

The if-else conditional statement enables you to execute a block of statements depending upon a condition. ELSEIF statements are not supported.

```
<c:if test="${a == 1 && b == 2}">
 ...
c:else/>
 ...
</c:if>
```

7.12.4.2 While Conditional Statement

The while statement enables you to execute a block of statements repeatedly until the condition check fails.

```
<c:while test="${a == 1}">
 ...
</c:while>
```

7.12.4.3 Foreach Conditional Statement

The `foreach` conditional statement enables to you enumerate built-in enumeration objects, such as `SQL Result Set` and `Vectors`. Also, this statement is used to execute a block of statements repeatedly by stepping through a `STEP` value.

```
<c:foreach var="row" items="${rs}">
 <c:out value="${row[0]}"/>
</c:foreach>

Then use the break statement to exit the loop:

<c:foreach var="row" items="${rs}">
 <c:if test="${row[0] begin="1" end="${count}" step="3">
   <c:out value="${row}"/>
</c:foreach>
```

7.12.4.4 Break Statement

Break from a loop with the `break` statement.

```
<c:foreach var="row" items="${rs}">
 <c:if test="${row[0] == "1"}">
   <c:break/>
</c:foreach>
```

7.12.4.5 Choose Statement

The `choose` statement supports a mutually exclusive conditional execution, where only one of a number of possible actions is executed. The following example executes one of the `when` blocks depending on `value`:

```
<c:choose>
  <c:when test="${value < 20}">
    <c:out value="Greater than 20"/>
  </c:when>

  <c:when test="${value == 20}">
    <c:out value="Equal to 20"/>
  </c:when>

  <c:otherwise>
    <c:out value="Less than 20"/>
  </c:otherwise>
</c:choose>
```

7.12.5 Define Custom Functions

You can define custom functions. These functions have a global scope from the point of definition, which means that they can access all global variables within the same OTL page. All variables defined within a function have local scope, except for the `Array` data type.

In the following example, the `par1`, `par2`, `par3`, and `local` variables have local scope. Any modifications to these variables are not relected in other parts of the script. If you want to return more than one object from a function, use the `System.SetAttribute` and `System.GetAttribute` methods.

```
<c:func var="PrintData" params="par1, par2, par3">
  <c:out value="${par1}"/>
  <c:set var="local" value="${par1}"/>
  <c:return value="${par2 + par1}"/>
```

```
</c:func>

<c:set var="a" value="${PrintData ("Function Call", 1, 2)}"/>
<c:out value="${a}"/>
```

7.12.6 Manage the Database Connection

You can use `database` to specify the database connection information used to establish a connection for the application or to disconnect from the database. Only one connection for each application is allowed in the OTL engine.

7.12.6.1 Specify Database Connection Information for an Application

Specify the database connection information used to establish a connection for the application. There is only one connection for each application

```
<c:database username="SYSTEM" password="P" DSN="POLITE""/>
```

7.12.6.2 Disconnect from the Database

To disconnect from the database, then issue the following:

```
<c:database action="disconnect"/>
```

7.12.7 Global Classes

The device manager OTL engine contains two predefined global classes, which are available to any script that access operating system and device manager information.

- Section 7.12.7.1, "Methods of the System Class": Use to access operating system information.
- Section 7.12.7.2, "Methods of the DeviceManager Class": Use to access device manager information.

7.12.7.1 Methods of the System Class

You can use the following system functions in your device manager command:

- Section 7.12.7.1.1, "Retrieve HTTP Request Parameters and Session Values"
- Section 7.12.7.1.2, "Create a Date Object"
- Section 7.12.7.1.3, "Create a Time Object"
- Section 7.12.7.1.4, "Get, Set, or Remove Session Attributes"
- Section 7.12.7.1.5, "Retrieving Parameter Name or Value"
- Section 7.12.7.1.6, "Retrieving the Request URL"
- Section 7.12.7.1.7, "Retrieving the Last Error Message"
- Section 7.12.7.1.8, "Retrieving System Memory Information"
- Section 7.12.7.1.9, "Retrieving Storage Information"
- Section 7.12.7.1.10, "URL Encoding a String"
- Section 7.12.7.1.11, "Opening a File"
- Section 7.12.7.1.12, "Synchronizing Databases"

7.12.7.1.1 Retrieve HTTP Request Parameters and Session Values You can retrieve the existing HTTP request parameters and HTTP Session values, as follows:

Retrieve HTTP Request Parameters

You can retrieve all of the HTTP request parameters, such as the URL and Form parameters. To retrieve a specific parameter value, prefix the parameter name with a colon—such as `:param_name`—or use the `GetParameterValue` function. All of the parameter values are preprocessed and the URLs are decoded by the tag language for you.

For example, if a URL is `c://my_app/index.html?my-Par=abcde`, then you can retrieve the parameter value in either of the following ways:

```
<c:out value="${my_par} />
```

or you can use the `GetParameterValues` function to retrieve all of the input parameters and then use `GetParameterValue` function to retrieve the value of each individual parameter, as follows:

```
<c:set var="rs" value="${System.GetParameterNames()}" />
<c:foreach var="r" items="${rs}">
 <BR>Parameter Name = <c:out value="${r} />
 Parameter Value = <c:out value="$System.GetParmaeterValue (r)}" />
</c:foreach>
```

For more information on `GetParameterValues` and `GetParameterValue`, see Section 7.12.7.1.5, "Retrieving Parameter Name or Value".

Retrieve HTTP Session Attributes

You can retrieve and store session attributes by prefixing the parameter name with a colon (`:name`) or through the Session get and set functions.

```
<c:set var="dummy" value="${System.SetAttribute ("NAME", "VALUE")}"/>
<c:set var="val1 value="${System.GetAttribute ("NAME")}"/>
<c:set var="val2 value="$(:NAME}"/>
```

As you can see, `val1` demonstrates how to retrieve the value using the `GetAttribute` function and `val2` demonstrates how to retrieve the value using `:NAME`. Substitute the actual HTTP session parameter name for `NAME`.

For more information on get and set attribute functions, see Section 7.12.7.1.4, "Get, Set, or Remove Session Attributes".

7.12.7.1.2 Create a Date Object Create a Date object with the current time using `System.Date()`. Create a Date object with a predefined date value using `System.Date(String date)`, where `date` is a date string.

7.12.7.1.3 Create a Time Object Create a Time object with the current time using `System.Time()`. Create a Time object with a predefined time value using `System.Time(String time)`, where `time` is a time string.

7.12.7.1.4 Get, Set, or Remove Session Attributes You can get, set, or remove Session attributes. To set an attribute in the application session, use `System.SetAttribute (String name, Object value)`. Each attribute has a unique name and value. To get the attribute value, use `System.GetAttribute (String name)`. To remove the attribute, use `System.RemoveAttribute (String name)`.

7.12.7.1.5 Retrieving Parameter Name or Value You can retrieve all of the parameters that are provided through the `GetParameterNames` method, which returns all parameters in an `Enumeration` object. Given a parameter name, you can retrieve the value through the `System.GetParameterValue(String name)` method.

Retrieve all parameters into the `params` variable, as follows:

```
<c:set var="params" value="${System.GetParameterNames()}" />
```

Then, once you have retrieved the parameters, you can parse through them using a for loop, as follows:

```
<c:foreach var="parm" items="${params}">
 <BR>Parameter Name = <c:out value="${parm} />
 Parameter Value = <c:out value="$System.GetParameterValue (parm)}" />
</c:foreach>
```

Each parameter is read into the `parm` variable and the name is retrieved using `value="${parm}`. The value is retrieved with the `System.GetParameterValue` method.

7.12.7.1.6 Retrieving the Request URL Use the `System.GetURL()` method for retrieving the request URL.

7.12.7.1.7 Retrieving the Last Error Message Use the `System.GetError` method for retrieving the last error message. If any of the command statements resulted in an error, such as a database error while executing a SQL statement, retrieve the error using this method.

7.12.7.1.8 Retrieving System Memory Information Use `GetMemoryInfo` method to retrieve the device memory information. The following parameters are supported by this function:

- 0 - Retrieve free memory (virtual)
- 1 - Retrieve total memory (virtual)
- 2 - Retrieve free memory (physical)
- 3 - Retrieve total memory (physical)

`System.GetMemoryInfo (Integer type)`—Given a value between 0 and 3, returns a `Long` value containing the requested memory information.

7.12.7.1.9 Retrieving Storage Information Use System.GetStorageInfo to retrieve device storage information. The return value sent back is in KB.

- 0 - Retrieve free storage
- 1 - Retrieve total storage

The second parameter must be a drive name or a directory name. If the function is invoked without parameters, then the function retrieves the free storage space in the root directory.

`System.GetStorageInfo(Integer type, String drive)`—returns a `Long` value containing storage information.

7.12.7.1.10 URL Encoding a String Encodes the provided string.

System.URLEncode (String value)

Returns the given string as a URL encoded string.

7.12.7.1.11 Opening a File Use `System.OpenFile (String name)` to open the file provided in the method parameter. Returns the `File` object.

7.12.7.1.12 Synchronizing Databases Use the `System.CreateSyncClient` method to create a `Synchronization` client object. Call the `Synchronization.Synchronize` method to synchronize. Retrieve any error messages using the `System.GetError` method.

```
<c:set var="sync " value="${System.CreateSyncClient()}"/>
<c:set value="${sync.SetUserName ("S11U1")}"/>
<c:set value="${sync.SetPassword ("manager")}"/>
<c:set value="${sync.SetServerURL ("http://localhost")}"/>
<c:set value="${sync.SetProxyInfo ("www-proxy:80")}"/>
<c:set var='ret' value="${sync.Synchronize()}"/>
<c:if test="${ret != 0}">
   <BR>Synchronization error = <c:out value="${System.GetError()}"/>
</c:if>
```

7.12.7.2 Methods of the DeviceManager Class

The device manager methods are accessed using `DeviceManager.FunctionName` syntax. These functions can only be used by trusted OTL scripts.

- DeviceManager.UploadFile (File file, String URL)
- DeviceManager.GetServerURL ( )
- DeviceManager.GetBinaryDir ( )
- DeviceManager.GetUserName ( )
- DeviceManager.CreateRequest (String cmd)
- DeviceManager.GetRegistry (String key, String name)
- DeviceManager.SetRegistry (String key, String name, String value)
- DeviceManager.LogMessage (String handler, String name, String message)

DeviceManager.UploadFile (File file, String URL)

Use `UploadFile` method to upload a file to the Mobile Server, which contains the device manager server. In order to use this method, you must first successfully use `System.OpenFile` on the file in question.

Given a `File` object and a URL, returns true if the upload is successful.

DeviceManager.GetServerURL ( )

Returns the URL of the Mobile Server.

```
<c:set var='url' value='${DeviceManager.GetServerURL()}'/>
```

DeviceManager.GetBinaryDir ( )

Returns the full path of the binary directory of Oracle Database Lite client.

```
<c:set var='dir' value='${DeviceManager.GetBinaryDir()}'/>
```

DeviceManager.GetUserName ( )

Returns the Oracle Database Lite username

```
<c:set var='user' value='${DeviceManager.GetUserName()}'/>
```

DeviceManager.CreateRequest (String cmd)

Create a Device Manager request (or command) and notify Device Manager Agent to process it. Command string must have corresponding OTL script file in the client device.

The following example demonstrates notifying the DM Agent to process and OTL script of the name: `sync.otl`.

```
<c:set value='${DeviceManager.CreateCommand ("sync")}'/>
```

DeviceManager.GetRegistry (String key, String name)

Retrieve a value from the Oracle Database Lite configuration file (`POLITE.INI`). All the values are retrieved as `String`. The following example retrieves the value for `Data_Directory` configured in the `POLITE.INI` file.

```
<c:set var='val' value='${DeviceManager.GetRegistry ("All Databases", "Data_
Directory")}'/>
```

DeviceManager.SetRegistry (String key, String name, String value)

Set a new configuration value in the Oracle Database Lite configuration file (`POLITE.INI`). The following example sets a new value for `Data_Directory`.

```
<c:set value='${DeviceManager.SetRegistry ("All Databases", "Data_Directory",
"C:\TEMP")}'/>
```

DeviceManager.LogMessage (String handler, String name, String message)

Log a message in the Device Manager logging system. The Device Manager client uploads all logged messages to the Mobile Server.

```
<c:set value='${DeviceManager.LogMessage (0, "My Log", "Log message..."}'/>
```

Applications may use this method to send data to the server. In order to accomplish this, you must create a handler on the server to process the client message. Once created, you must register this handler in the Mobile Server. If your application is written in C/C++, JAVA or Visual Basic, you may use corresponding native APIs to log any message. For a C/C++ application, use the following:

```
dmLogMessage (const TCHAR* handler, const TCHAR* name, const TCHAR* message);
```

In order to use the above API, you must dynamically load the `OMCAPI.DLL` library and extract the function pointers. The following sample code demonstrates how to log a message:

```
typedef void (*dm_Initialize)();
typedef void (*dm_Destroy)();
typedef void (*dm_LogMessage) (LPCTSTR, LPCTSTR, LPCTSTR);
HMODULE hMod = ::LoadLibrary (TEXT ("omcapi.dll"));
if (hMod)
{
  dm_Initialize init =
         (dm_Initialize)::GetProcAddress (hMod, TEXT ("dmInitialize"));
  dm_Destroy    dest = (dm_Destroy)::GetProcAddress (hMod, TEXT ("dmDestroy"));
  dm_LogMessage log  =
         (dm_LogMessage)::GetProcAddress (hMod, TEXT ("dmLogMessage"));
  if (init && dest && log)
  {
    (*init)();
```

```
    (*log) (TEXT ("MY_HANDLER"), TEXT ("MY LOG"), TEXT ("My Message"));
    (*dest) ();
   }
   ::FreeLibrary (hMod);
}
```

If you want to use the Java API, then set `WEBTOGO.JAR`, which is part of 'Oracle Lite WEB' client. The following code sample demonstrates how to log a message with Java APIs:

```
import oracle.lite.dm.ClientAPI;
public class MyLog
{
  public static void main (String[] args)
  {
    ClientAPI.initialize();
    ClientAPI.logMessage ("MY_HANDLER", "MY LOG", "My Message");
    ClientAPI.destroy();
  }
}
```

On the server-side, a message handler must be developed and registered. A message handler is a Java class that implements the `oracle.lite.provider.MessageListener` interface. See the Javadoc for more information on the `MessageListener` interface. The following is an example of a message handler:

```
import oracle.lite.resource.Device;
import oracle.lite.provider.MessageListener;
import oracle.lite.provider.MessageData;
public class MyMessage implements MessageListener
{
  public void initialize (String metaData) throws Exception {}
  public void destroy() throws Exception {}
  public void service (Device device, String name, MessageData data)
           throws Exception
  {
    // Process 'My Message'
   }
}
```

Once the message handler is implemented and compiled, copy the JAR file to *<ORACLE_HOME>*`\Mobile\class` directory. Then, execute the following SQL script to register your message handler implementation:

```
msql mobileadmin/manager@jdbc:oracle:thin:@<mobile_server_db>:<port>:<sid>
insert into dm$all_providers values ('MY_HANDLER', 'MESSAGE', 'MyMessage', NULL);
```

7.12.8 Importing Another OTL Page

Use the `import` statement to include a page into the current page. All URL parameters are available to scripts in the imported page, as well as in the current page below the point of inclusion.

```
<c:import url="url_of_the_include_page"/>
```

Specify URL parameters using the HTTP format with the `?` or the `<c:param>` tag, as shown below:

```
<c:import url="URL?abc=def">
  <c:param name="name1" value="value1"/>
```

```
  <c:param name="name2"> value2 </c:param>
</c:import>
```

7.12.9 Error Handling

You can throw and catch exceptions within any OTL script. The only restriction is that you can only have at most one catch block. The throw script terminates the current script processing and jumps to the catch block. If a page contains a single throw tag and no catch tag, then the script stops upon reaching the throw tag.

```
<c:throw value='1'/>
...
<c:catch var='ex'/>
...
</c:catch/>
```

7.12.10 Sample Device Manager Commands Using the Tag Language

Retrieve current date and time:

```
<c:set var="d" value="${System.Date()}"/>
<c:out value="Date = ${d}"/>
<br><c:out value="Time = ${d.Time()}"/>
```

Perform Date arithmetic by first retrieving the date, then adding or subtracting 2 days from it. The last two lines changes the date to either be 2 days from now or 2 days ago.

```
<c:set var="d"  value="${System.Date()}"/>
<c:set var="d2" value="${d + 2}"/>
<c:set var="d3" value="${d - 2}"/>
Date + 2 = <c:out value="${d2}"/>
<br>Date - 2 = <c:out value="${d3}"/>
```

Formatting date and time by retrieving the time using the `Date` or `Time` methods, and then applying a format. For the date, apply either day/month/year or month/day/year with the `Format` method. For time, you can choose the format of hours:minutes.

```
<c:set var="d"  value="${System.Date()}"/>
Date (dd/mm/yyyy) = <c:out value="${d.Format ("dd/mm/yyyy")}"/>
Date (mm/dd/yyyy) = <c:out value="${d.Format ("mm/dd/yyyy")}"/>
<c:set var="t" value="${System.Time()}"/>
Time (hh:mm) = <c:out value="${t.Format ('hh:mm')}"/>
```

You can apply the names of the month or day to a date by using `GetMonth` and `GetDay`, as follows:

```
<c:set var="month" value="${{"Jan", "Feb", "Mar", "Apr", "May", "Jun",
            "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"}}"/>
<c:set var="day" value="${{"Sunday", "Monday", "Tuesday", "Wednessday",
            "Thursday", "Friday", "Saturday"}}"/>
<c:out value="${month[d.GetMonth() - 1]}"/>
<c:out value="${day[d.GetDay()]}"/>
```

Retrieve the day and add two days to it:

```
<c:set var="d" value="${d + 2}"/>
```

Set the date to 02-20-2004.

```
<c:set var="d" value="${date ("02-20-2002")}"/>
```

8 Manage Your Branch Office

The following sections describe how to install, configure, manage and use the Mobile client for Branch Offices:

- Section 8.1, "Introduction"
- Section 8.2, "Branch Office Installation and Configuration"
- Section 8.3, "Architecture"
- Section 8.4, "Administration"

> **Note:** If you have installed Branch Office 10g Release 1 and want to use a later version, you must perform some upgrade steps that are listed in the Upgrade chapter of the *Oracle Database Lite Getting Started Guide*.

8.1 Introduction

The following sections introduce the Oracle Database Lite Branch Office:

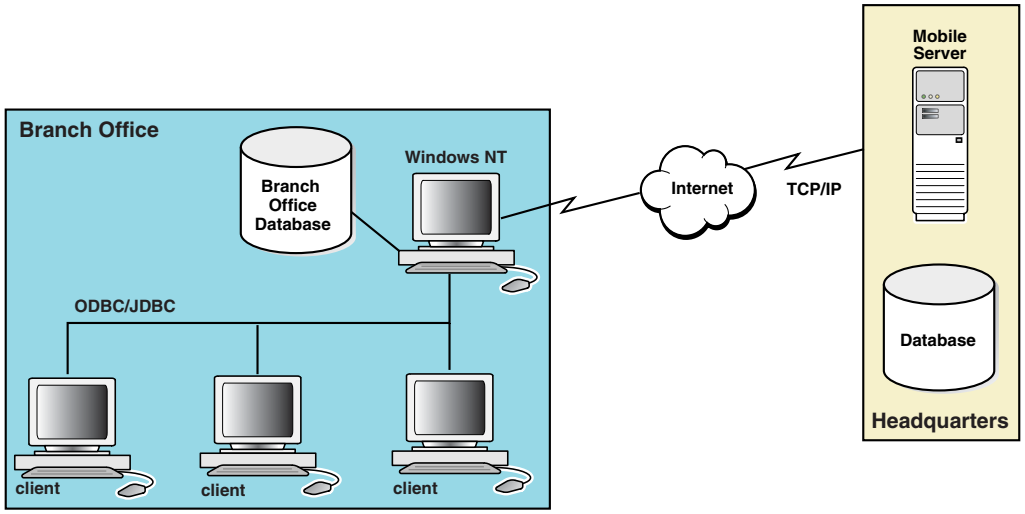
- Section 8.1.1, "What is the Branch Office?"
- Section 8.1.2, "How the Branch Office Works"
- Section 8.1.3, "The Branch Office Manager"
- Section 8.1.4, "Synchronizing Data with Headquarters"

8.1.1 What is the Branch Office?

The Branch Office provides access to the Branch Office database for up to 32 concurrent networked users. It enables the deployment of enterprise data and applications to geographically distributed sites that are running a Branch Office database. Each Branch Office database is centrally managed and supports multiple client connections, thereby eliminating local database administration tasks.

The Branch Office database synchronizes client data with the Oracle database at the company headquarters. Figure 8–1 illustrates the Branch Office database at a Branch Office location and its connection to the Oracle database server at the headquarters. Branch Office clients connect to the Branch Office database using either ODBC or JDBC connections. Clients access and update the Branch Office database, which contains a subset of the corporate database located at the company headquarters.

Figure 8–1 A Branch Office Overview



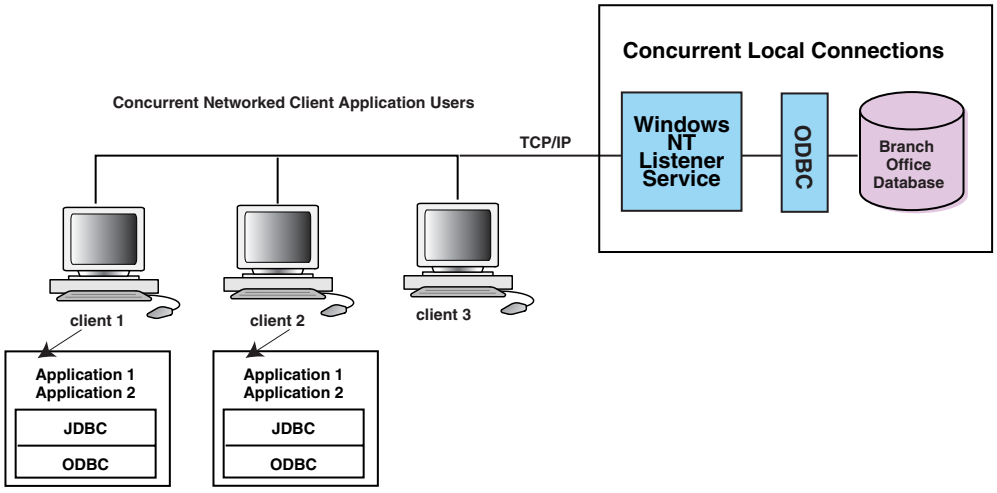
8.1.2 How the Branch Office Works

Each Branch Office database supports up to 32 concurrent networked users, which are also known as Branch Office clients. These clients do not require a connection to their company headquarters and are allowed to work independently, without the corporate database.

The Branch Office also supports 32 concurrent local ODBC/JDBC connections to the Branch Office database. These local connections can be used for applications that perform background tasks, such as reporting, mass changes or updates, and bulk data loading.

As Figure 8–2 displays, Branch Office clients and local ODBC and JDBC applications can access the Branch Office database simultaneously. Multiple applications can execute on each client.

Figure 8–2 Accessing the Branch Office Database



8.1.3 The Branch Office Manager

The Branch Office requires no local database administration and enables configuration and monitoring of Branch Office database services and users. Using the Web-based Branch Office Mobile Manager interface, the Branch Office Administrator centrally manages Branch Office operations. Furthermore, by using the Branch Office Mobile Manager, the administrator does not need to be physically present at each Branch Office.

8.1.4 Synchronizing Data with Headquarters

Data synchronization for Branch Office is centrally managed by the Branch Office Administrator. The Administrator synchronizes applications and data with the database located at headquarters through a TCP/IP connection. Synchronization between the Branch Office database and the headquarters database is executed through the Mobile Server. For more information on synchronizing data, see Chapter 5, "Managing Synchronization" and the Chapter 3 "Synchronization" in the *Oracle Database Lite Developer's Guide*.

The centralized management and data synchronization between the headquarters and its branches enables each Branch Office to synchronize data with the corporate database according to a pre-determined schedule. This allows for data replication based on geographic factors and alternate time zones.

Data specific to a given Branch Office is synchronized from the corporate database server to the Branch Office database. Each Branch Office database represents a single instance of replicated data and is the physical data repository that is accessed by Branch Office clients.

8.2 Branch Office Installation and Configuration

The following sections describe how to install and configure the Mobile client for Branch Office:

- Section 8.2.1, "Terms and Concepts"
- Section 8.2.2, "Overview"
- Section 8.2.3, "Branch Office Pre-Installation Considerations"
- Section 8.2.4, "Branch Office Installation"
- Section 8.2.5, "Enabling Branch Office on Windows XP Service Pack 2"
- Section 8.2.6, "Changing Branch Office Listener Port Number and Working Directory"
- Section 8.2.7, "Accessing Branch Office or the Multi-User Service Using an ODBC or JDBC Driver"

8.2.1 Terms and Concepts

Branch Office

A deployment concept of Oracle Database Lite designed for remote offices and Branch Office configuration.

Mobile client for Branch Office

Self-contained bundle of Oracle Database Lite libraries installed in the maching that contains the Branch Office.

Branch Office Administrator

Logical user responsible for the management of Branch Office users, data, and applications.

Branch Office Database

Multi-user of the Oracle Lite database.

Branch Office Application

Native or Java applications that access the Branch Office database over remote ODBC or JDBC connections.

Branch Office User

Logical user who is a client of the Branch Office database.

Branch Office Administrators

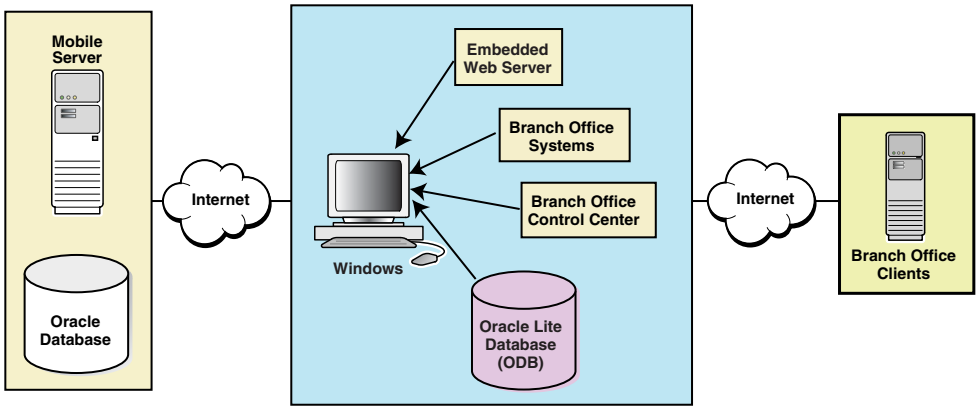
Group of Branch Office Administrators, each managed by the Mobile Manager.

8.2.2 Overview

To help understand and successfully implement a Branch Office setup, this section presents a sample setup that simulates a typical Branch Office environment. As Figure 8–3 displays, this example assumes that the Branch Office configuration has the following installations on three machines.

1. M 1: One Mobile Server with a corporate Oracle database server.
2. M 2: One or many Branch Office system(s) running on a Windows machine. This setup includes an embedded Web server, a multi user Oracle Lite database, and the Branch Office Mobile Manager. The Branch Office libraries are installed as part of the Mobile client for Branch Office.
3. M 3: Up to the maximum of thirty-two Branch Office clients that host the Branch Office application (`.exe`) and use a remote ODBC/JDBC connection to access data in the Branch Office (multi-user) database located on M 2.

Figure 8–3 Branch Office Setup



8.2.3 Branch Office Pre-Installation Considerations

When you install the Branch Office Manager on the Windows machine, it creates the `OracleDatabaseLite` user account with the minimum set of privileges required to execute the Oracle Database Lite software. This prevents Oracle Database Lite Branch Office executing under the `SYSTEM` account, which has broad privileges within the system and can make the system vulnerable.

Both the 'Oracle Lite Multiuser Service' is created as well as the normal Web-to-Go service executes under the privileges of the `OracleDatabaseLite` user. The Oracle Lite Multiuser Server enables remote clients to connect to the Oracle Lite database.

Normally, when installed, the password for the `OracleDatabaseLite` user is randomly generated during the setup. You can either pre-configure this password before the Branch Office installation or modify it after the configuration. See Section 3.5.3, "Defining Password for OracleDatabaseLite User for Branch Office on Windows Machine" in the *Oracle Database Lite Getting Started Guide*.

8.2.4 Branch Office Installation

To install and configure the Branch Office, perform the following steps:

1. Install the Mobile Server on the machine named M1.
2. Using the Packaging Wizard, package the Branch Office application. During the application packaging process, select *Oracle Lite Branch Office* as your target platform. For more information on how to package your applications using the Packaging Wizard, refer to the Packaging Wizard chapter in the *Oracle Database Lite Developer's Guide*.
3. Using the Mobile Manager Applications page, publish your Branch Office application. Select the Branch Office application that you need to publish and click **Publish Application**.
4. Using the Mobile Manager Users page, create a Branch Office Administrator user and add this user to the *Branch Administrators* group. Provide administrator privileges to the Branch Office Administrator user.
5. Using the Mobile Manager Applications page, click the published Branch Office application link. Select the Files tab and choose the application files that you want installed on the Branch Office Client (M3) machine. Click **Make Public** to specify that these files are public files.
6. Provide access privileges to the Branch Office Administrator user. Using the Mobile Manager Applications page, select the Branch Office application that you need to provide user access privileges for and click **Access**. Grant access privileges by selecting the check box displayed against the Branch Office Administrator.
7. Using the following URL, download and install the Mobile client for Branch Office onto the Branch Office machine (M2).

   ```
   http://<mobileserver>/webtogo/setup
   ```

8. Using the following URL, open a browser window in the Branch Office machine (M2) and connect to the local Branch Office Web Server using the appropriate Branch Office Administrator user name and password:

   ```
   http://<branch_office_hostname>
   ```

   OR

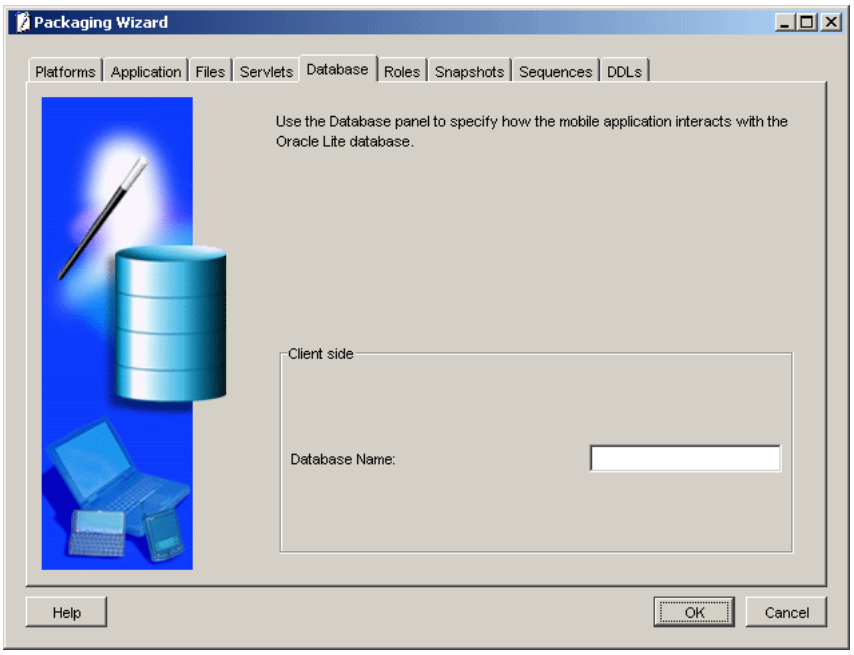
   ```
   http://localhost
   ```

Note: Normally the Branch Office Web Server is automatically started by the setup program. If not, open the Control Panel and choose Services. Start the service name *Oracle Web-to-Go*.

9. At this stage, the Branch Office performs a complete synchronization process with the Mobile Server.

   a. A directory is created under the `oldb40` directory with the name of your Branch Office Administrator user name. Under this directory, the Branch Office creates a database file with the same name as the **Database Name** provided in the Packaging Wizard as displayed in Figure 8–4.

Figure 8–4 Database Panel - Packaging Wizard



   b. The Branch Office automatically creates a DSN entry in a `<user name>_<database name>` format.

      For example, if your Branch Office Administrator user name is `Tom` and your database name is `BranchDB`, the Branch Office creates a DSN entry named `Tom_BranchDB`.

10. Open the Branch Office Mobile Manager and create a new Branch Office user.

    This stage marks the conclusion of the Branch Office Installation and Configuration process. In the next step, you must configure the Branch Office Client (M3).

11. On the Branch Office Client machine (M3), open a browser window using the following URL and download the *ODBC Driver* program. This action creates a DSN entry in the Branch Office Client machine (M3) with the same name as the **Database Name** provided in the Packaging Wizard.

    ```
    http://<branch_office_hostname>/public/download
    ```

Note: The DSN name on the Branch Office Client machine (M3) created by the ODBC Driver program is different from the DSN name on the Branch Office machine (M2).

12. On the Branch Office Client machine (M3), open a browser and download the Branch Office application files using the following URL.

    ```
    http://<branch_office_hostname>/public/download
    ```

13. On the Branch Office machine M3, add `oladc12040.dll` and `olc12040.dll` to your path.

14. If the client created in step 11 is `boUser`, then when you connect from the client to the Branch Office, the remote JDBC connection string for the user named `boUser` is shown below.

    ```
    <boUser>/<boUser password>@jdbc:odbc:<DSN>
    ```

    The DSN name is the same as the name provided in step 9 in the `<user name>_<database name>` format in the Packaging Wizard. It can be located in the `ODBC.INI` file on the Branch Office Client machine (M3). The DSN points to the remote database listener located on the Branch Office machine (M2). The default port number for the database listener can be modified by modifying the port number in the `SERVICE_PORT` parameter in the `POLITE.INI` file. The default working directory can be modified with the `SERVICE_WDIR` parameter in the `POLITE.INI` file.

8.2.5 Enabling Branch Office on Windows XP Service Pack 2

To enable Branch Office on Windows XP Service Pack 2, you need to perform the following:

When you install Windows XP Service Pack 2, the ICF defaults to ON. In order for the Branch Office Server to work properly, you need to enable the Web-to-Go service and Branch Office executables access through the firewall. You can add these executables, as follows:

Go to the Windows Firewall control on your Windows machine. Select the Exception tab. Click **Add Program**. Browse for the following two programs and select the appropriate executables. Click **OK**.

- `%MOBILE_CLIENT_INSTALL%\bin\wtgsvc.exe`
- `%MOBILE_CLIENT_INSTALL%\bin\olsv2040.exe`

8.2.6 Changing Branch Office Listener Port Number and Working Directory

In order to change the Branch Office Server Listener port number or working directory, perform the following steps:

1. Stop both the '**Oracle Web-to-go**' and the '**Oracle Lite Multiuser**' services.

2. In the `polite.ini` file, edit the `[All Databases]` section to include the `SERVICE_PORT` or `SERVICE_WDIR` parameter, which points to the new listening port.

   For example:

   ```
   [All Databases]
   SERVICE_PORT=1160
   ```

```
SERVICE_WDIR=C:\WINDOWS\SYSTEM32
```

3. Start the '**Oracle Lite Multiuser**' service first and then the '**Oracle Web-to-go**' service.

> **Note:** The sequence in which services are started and stopped should be in the order as described above.

8.2.7 Accessing Branch Office or the Multi-User Service Using an ODBC or JDBC Driver

In order to access a Branch Office or the Multi-User Service—using either the ODBC or JDBC drivers—the Branch Office or Multi-User Service host where these reside must define a DSN for the host within the `ODBC.INI` file. This DSN is used by the remote clients to access the Branch Office or Multi-User service.

On the client, you can define the host where the Branch Office or Multi-User service resides in the URL with either the following:

- Specify the DSN name in the URL.
- If you specify `NONE` as the name of the DSN, then specify the `Database` and `DataDirectory` in the connection string where the values are the same as one of the DSNs in the `ODBC.INI` file.
- If you have specified the `Database` or `DataDirectory` attributes in the connection string for type 2 or type 4 driver, then the value for either of the two attributes must be the same as the one defined in the `ODBC.INI` file, otherwise, the connection is rejected.

8.2.8 Changing the Language or Locale for Branch Office Client

If the user needs to change the locale for the default user-profile for a Branch Office client running in service mode, then perform the following:

If the Branch Office client is installed on Windows XP, then perform the following:

1. Log on to the computer as the administrator.
2. Open the "Regional and Language Options" in the Control Panel.
3. Select the Advanced tab and select the Default user settings checkbox.
4. Click Apply and restart the computer.

If the Branch Office client is installed on Windows 2000, then perform the following:

1. Log on to the computer as the administrator, and then create a local user account.
2. Log off as the administrator, and then log on to through the local user account that you just created.
3. Change the locale of user to the desired locale within the **Control-Panel->Regional Settings** page.
4. Log off as the local user, and then log back on as the administrator.
5. Turn on the following option: **Show hidden files and folders**. In Windows Explorer, this option can be selected in the View tab of the Tools->Folder Options screen.

6. Replace the current default user profile with the customized default user profile, as follows:
   a. Navigate to the Control-Panel->System.
   b. On the User Profiles tab, click the user profile that you just created, and then click **Copy To**.
   c. In the Copy profile to section, select the location and who is permitted to use this profile. Click **Browse** and select the `\Documents and Settings\Default User` folder for where the profile is to be copied. To set the permissions, then under the **Permitted to use** section, click **Change** for **everyone**. Click **OK** to save.

This modifies the locale for the default user-profile. At this point, you should install the Branch Office, which will reflect the new locale.

8.3 Architecture

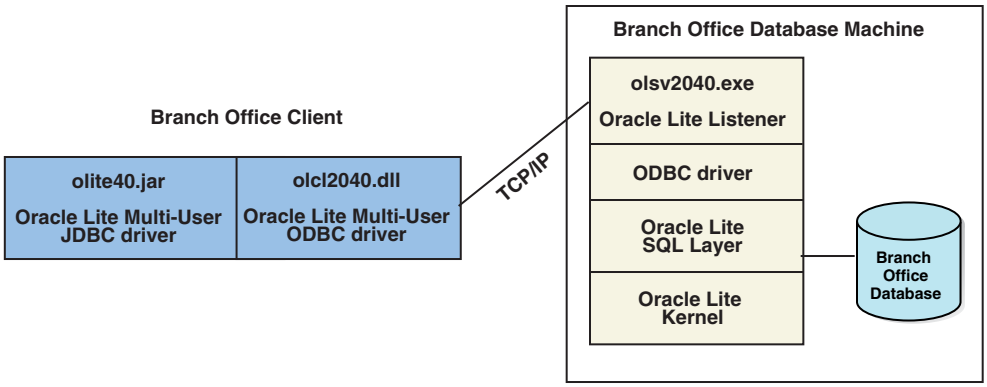
The following sections describe the components of the multi-user architecture:

- Section 8.3.1, "The Branch Office Environment"
- Section 8.3.2, "Connecting Clients to the Branch Office Database Machine"

8.3.1 The Branch Office Environment

The Branch Office environment is comprised of two parts. As Figure 8–5 displays, they are the Branch Office Client component and the Branch Office Database component.

Figure 8–5 The Branch Office Environment



8.3.1.1 The Branch Office Client

The Branch Office client machine executes both ODBC and Java based applications which access the Branch Office database (`branch.odb`).

> **Note:** The `branch.odb` file represents a sample database used throughout this chapter. Your database name may be different.

The Branch Office includes the following components.

- ODBC Driver
- JDBC Driver

ODBC Driver

The client ODBC driver (`olcl2040.dll`), supports ODBC-based client application connections for the Branch Office database. This driver connects ODBC applications to the Branch Office database. Based on the parameters specified in the client DSN, it searches for the Windows NT service running on the Branch Office database machine.

JDBC Driver

Java-based client applications connect to the Branch Office database through a JDBC connection. This JDBC driver (*ORACLE_HOME*`/Mobile/Sdk/bin/olite40.jar`) uses the ODBC driver for the connection. The ODBC driver makes the actual connection for the JDBC client application by first reading the DSN defined parameters and then by searching for the associated Windows NT service.

8.3.1.2 The Branch Office

The Branch Office contains the following components.

- Branch Office Database
- Oracle Database Lite Listener Service
- Mobile client for OC4J or Web-to-Go

Branch Office Database

The Branch Office database machine is the interface between Branch Office clients and the database at company headquarters.

The Branch Office database (`branch.odb`) is a file created by the Mobile Server during synchronization. This database file is a subset of the headquarters database. Its tables are built on the headquarters database server. The Branch Office database file does not support Oracle Database Lite utilities, such as `CREATEDB` or `REMOVEDB`.

The Mobile Server Packaging Wizard defines and generates replication support for tables. For more information, see Chapter 6 "Using the Packaging Wizard" in the *Oracle Database Lite Developer's Guide*.

Note: Snapshots are owned by `SYSTEM`. The password is the Branch Office administrator password.

Oracle Database Lite Listener Service

The Branch Office database machine contains the ODBC listener service named `olsv2040.exe`. This process creates a separate connection to the Branch Office database for every client network connection.

The listener service is dependent on Java. Before starting the listener service, the database machine must have the Sun Microsystems Java Runtime Environment (JRE). The JRE can be downloaded from the Java technology Web site.

The system `PATH` variable must include a path reference to this `bin` directory.

Mobile client for OC4J or Web-to-Go

The Mobile client for OC4J or Web-to-Go platform executes on the Branch Office database machine and acts as a Web server to run the Branch Office Mobile Manager. This feature allows the system administrator to access the Branch Office Mobile Manager and maintain a Branch Office database without being physically present at the Branch Office.

The Mobile client for OC4J or Web-to-Go enables users to deploy applications on client machines, using a browser that points to the Branch Office database machine. The Mobile client for OC4J or Web-to-Go publishes client applications as public files, so that client applications that use the Branch Office database can be downloaded directly from the Branch Office database machine. The Mobile client for OC4J or Web-to-Go executes as a background process to support browser based applications and distribution of public files.

8.3.1.3 Company Headquarters

The Oracle database resides at the company headquarters. The Mobile client for OC4J or Web-to-Go platform on the Branch Office database machine synchronizes all data changes in the Branch Office database with the Oracle database located at the headquarters.

8.3.2 Connecting Clients to the Branch Office Database Machine

The client applications connect with the Branch Office database machine through TCP/IP. The client driver, `olcl2040.dll`, facilitates this communication by connecting with the `olsv2040.exe` listener service on the Branch Office database machine. For every client connection, the listener service establishes a separate connection thread with the Branch Office database, `branch.odb`.

Establishing concurrent client connections requires that the listener service on the Branch Office database machine be started before the network connections are established.

8.3.2.1 ODBC Connection

To make a client connection to a Branch Office database, you must first set up an ODBC data source name (DSN) using the ODBC Administrator.

To connect an ODBC client application to a Branch Office database, an application must create a connection to the database. For example,

```
"UID=SYSTEM;PWD=MANAGER;DSN=POLITECL;DATABASE=BRANCH"
```

Table 8–1 describes the above database connection statement.

Table 8–1 Database Connection Statement Description

Parameter	Description
UID	A valid database user.
PWD	A valid password to the database.
DSN	A data source name set up using the ODBC Administrator.
Database	The name of the local Branch Office database residing in the `OLDB40/<username>` folder in the *`ORACLE_HOME`* directory.

8.3.2.2 JDBC Connections

JDBC client applications make connections to the Branch Office database machine as given below.

```
Connect con=Drivermanager.getConnection (JDBC URL,user,password)
```

Table 8–2 describes the above Branch Office database connection statement.

Table 8–2 Branch Office Database Connection Description

Parameter	Description
JDBC URL	The database URL. For example, `jdbc:polite@database_host_name:port_number:DSN`
User	A valid database user.
Password	A valid password for the database.

Given below is a Java sample that describes connection for multiple users.

```
Connection conn = null

try
{

   Class.forName("oracle.lite.poljdbc.POLJDBCDriver");
                conn = DriverManager.getConnection
               ("jdbc:Polite@DATA_SERVER:1160:POLITECL","SYSTEM","MANAGER");
}
   catch(Exection e)
{

   System.out.println("An error has occurred.");
   System.out.println("Error accessing the Multi-user database");
   System.out.println(e)
   System.exit(0);
}
```

The listener service must be started either manually or automatically before network connections can be established. The listener service can be started through the services application in the Control Panel, or through the Branch Office Mobile Manager.

8.4 Administration

The Administration facility is a Web-to-Go browser based application that enables the Branch Office Administrator to monitor and configure Branch Office database services and users. Navigate to the Mobile Workspace to administer the Branch Office applications and replication jobs. The Branch Office Administrator is a Mobile Server user created by the Mobile Server Administrator and must be included as a member of the group, "BRANCH ADMINISTRATORS."

The Administration facility enables user information maintenance capabilities for the Branch Office Administrator to centrally manage user access privileges to the Branch Office database. The Administration facility supports the following user management tasks.

The following sections provide instructions for using the Branch Office Mobile Manager:

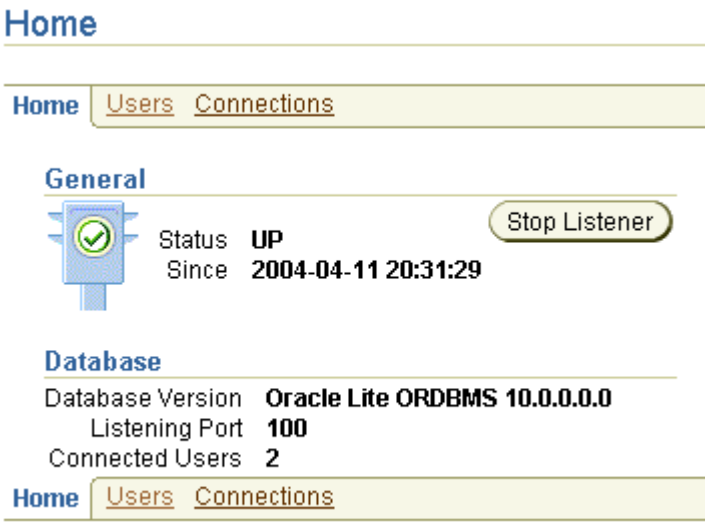
- Section 8.4.1, "Logging into the Branch Office Manager"
- Section 8.4.2, "Using the Branch Office Manager"
- Section 8.4.3, "Managing Branch Office Users"
- Section 8.4.4, "Managing Applications"

8.4.1 Logging into the Branch Office Manager

The Branch Office Administrator can access the Branch Office Manager by clicking the Branch Office Manager link in the workspace.

The Branch Office Manager appears and defaults to the Branch Office Home page, as displayed in Figure 8–6.

Figure 8–6 Branch Office Home Page



8.4.2 Using the Branch Office Manager

The Branch Office Manager contains the following pages. It enables the Branch Office Administrator to perform the administrative tasks described below.

- Home - The Branch Office home page enables you to start and stop the listener service. It displays general information such as system status, system details such as the Java version and operating system, and database information such as version, listening port, and number of connected users.
- Users - The Users page enables you to find and add users to the required database.
- Connections - The Connections page displays connection details such as user name, connection duration, and the database path.

8.4.2.1 Updating Status Summary

The General section of the Branch Office home page provides the listener status to the Branch Office Administrator. The listener status can be changed by starting or stopping the database service. The database section displays the latest status of the Branch Office database. To update the Branch Office status summary, click the Refresh button on your browser. Starting or stopping the listener service also updates the status summary.

8.4.2.2 Starting the Database Service

The Branch Office Manager home page enables the Administrator to start the Windows NT Listener service using the Start Listener button.

8.4.2.3 Stopping the Database Service

The Branch Office home page enables the Administrator to stop the Windows NT Listener service using the Stop Listener button.

> **Note:** A Branch Office Administrator should check the Status Summary for connected users before stopping the service. Local database connections are not detected by the Branch Office Mobile Manager.

8.4.2.4 Viewing the Status of the Branch Office Database

The Branch Office Manager supports an unlimited number of database files. The General and Database section in the Branch Office Manager home page enable an Administrator to view the status of the Branch Office database and start or stop the windows service. The Connection page displays additional database information.

Table 8–3 describes the Branch Office home page.

Table 8–3 Branch Office Home Page Description

Label	Function
Status	Branch Office status.
Since	Date and time since the Oracle Database Lite Branch Office system is up.
Database Version	Version number of the Oracle Database Lite Branch Office database.
Listening Port	The server port number that the Oracle Database Lite listener service uses.
Connected Users	Number of currently connected users.
Java Version	Version number of the Java Development Kit.
Operating System	Current operating system.

8.4.3 Managing Branch Office Users

To manage Branch Office users, login to the Mobile Server and navigate to the Users page. As Figure 8–7 displays, the Users page appears.

Figure 8–7 Branch Office Users Page

The Branch Office database does not need to be stopped to manage users and their access privileges. The Branch Office Administrator can add or delete users while other users are accessing the Branch Office database.

8.4.3.1 Creating Users

To create users, navigate to the Users page and click the Add User button under the Users section. The Add Users page appears. Enter the appropriate data in the corresponding fields and click the Save button.

Note: You should not create a user named "System." This user name is reserved for Web-to-Go use.

8.4.3.2 Setting User Roles

After you create a new user, the Branch Office Mobile Manager automatically displays the Roles page. Using the Roles page, the Branch Office Administrator can assign user roles by selecting the available role boxes. As Table 8–4 describes, the Branch Office Administrator can assign the following roles.

Table 8–4 User Roles Description

Field	Description
DBA	Database administrator privileges. When selected, users can add or remove users and add files to the database.
RESOURCE	RESOURCE privileges. Selecting this check box enables users to create their own sets of tables and relate them to their own schema.

8.4.3.3 Setting User Properties

The Roles Home page enables the Administrator to set user properties. To set user properties, click the Roles Home page link. Using this page, you can modify a user's password.

8.4.3.4 Setting User Privileges

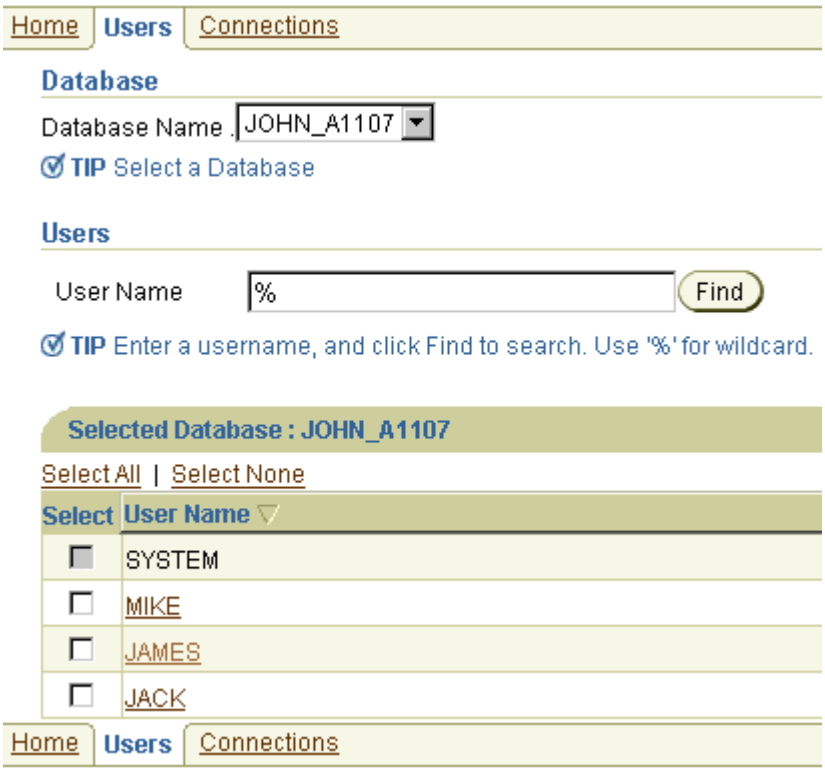
The Privileges page enables the Branch Office Administrator to assign user privileges. To control user access to database tables, you can grant user privileges such as Select, Delete, Insert, and Update.

8.4.3.5 Finding Users

To find all users, click the Users link. Select the appropriate database name and click the Find button. To find a specific user, enter the user name and click the Find button.

To display a list of all users for the chosen database, enter the % sign and click the Find button. As Figure 8–8 displays, the Users page displays users that are associated with the chosen database.

Figure 8–8 Displaying Branch Office Users



8.4.3.6 Removing a User

To remove a user, select the check box displayed against a user name and click the Delete button.

8.4.4 Managing Applications

The Applications tab enables the Branch Office Administrator to list all Web-to-Go applications that the Branch Office Administrator can access. Clicking an application link displays a list of files that comprise the application. The Branch Office Administrator can designate certain files as public, which means that they can be viewed and downloaded by the end users.

- Section 8.4.4.1, "Downloading Public Files to Your Client"

8.4.4.1 Downloading Public Files to Your Client

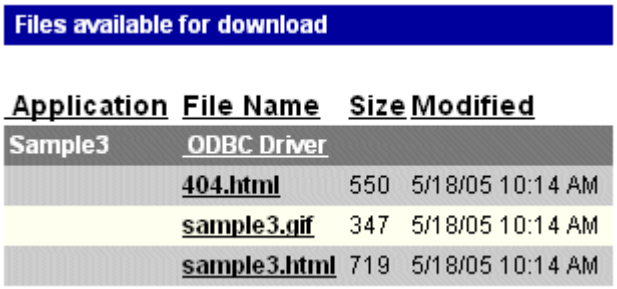
Making application files public enables the Branch Office to download those files and install them on the Branch Office clients.

To download and install a public file, Branch Office users must access the URL given below.

`http://<branchofficemachine>/public/download`

This URL lists all public files under their respective applications, as Figure 8–9 shows. Users can click the required file name and save it in their file system. After saving the file, users can install the application by running the self-extracting file.

Figure 8–9 Listing of Public Files



For ODBC configuration, click the ODBC driver link. This downloads the `setup.exe`. After the file is downloaded, users must run the `setup.exe`.

9 Offline Instantiation

The Offline Instantiation (OLI) utility enables Mobile administrators to prepare a package that includes the Mobile client software and initial data for every Mobile user in batch mode. This package can be used to set up a Mobile client with user-specific initial data within Oracle Database Lite. This procedure helps users avoid an expensive initial synchronization download to the Mobile client.

The following sections discuss the Offline Instantiation feature.

9.1 Using Offline Instantiation to Distribute Multiple Mobile Clients

You can enable your users to install their client using a distribution method, such as a CD, through the network, or email. To install the Mobile client and perform the first synchronization with the initial data can be a performance issue. In this case, the administrator can pre-create either just the Mobile binaries or the Mobile binaries with the user ODB files (includes the data for the user) for the client. The download of this package is faster than having each user perform the first synchronization on their device. Thus, this procedure helps users avoid an expensive performance hit when creating and synchronizing the Mobile client for the first time.

Offline instantiation is a tool that enables an administrator to gather and package the Mobile client binaries into a single directory. Offline instantiation is part of the Mobile Development Kit, which can be installed only on a Windows platform. Thus, you create all of your user distribution files on a Windows machine and you can only create multiple user distribution files for OC4J, Web-to-Go, Branch Office, Win32, and WinCE Mobile clients. We recommend that you use the same Windows environment where a Mobile server exists to create your distribution files.

When you have multiple users who use the same application, you set up a distribution for each user through the following steps:

1. Using the Mobile Manager on the Mobile Server, the administrator sets up the application and the users for the Mobile client distribution, as follows:
   a. Using the Mobile Manager on the Mobile Server, the administrator publishes the applications that are to be installed on the Mobile clients.
   b. The administrator creates all of the users for the Mobile clients.
   c. The administrator grants access for these users to the applications that are to be downloaded for the distribution.
2. On the Windows machine where the Mobile Development Kit is installed, perform a normal client installation and synchronization for one of your users as a template for the client distribution. Choose the platform and language that are appropriate for the Mobile clients that you are creating, except for the WinCE platform. To set up the Mobile client on the WinCE (PocketPC) device, choose the Oracle Lite WIN32. For steps on creating a Mobile client, see Section 4.1, "Preparing the Device for a Mobile Application" and Section 4.2, "Installing the Mobile Client" in the *Oracle Database Lite Getting Started Guide*.
3. Download the Mobile client binaries—without the application or user data—in a unique directory. See Section 9.3, "Downloading the Client Binaries Into the Client Root Directory" for directions.
4. Configure the `oli.ini` file to tell the offline instantiation batch tool how to create the client distribution packages. See Section 9.5, "Configure How OLI Creates the Client Distribution Packages With the OLI Configuration File" for directions.
5. Use the offline instantiation tool (OLI) to create and package the final client distribution packages for each user. See Section 9.6, "Using the OLI Engine to Create and Package the Client Distribution File" for directions.
6. Distribute the client distribution packages to each user to install on their client device. See Section 9.7, "Deploying Client Distribution Files on Client Machines" for directions.

9.2 Setting Up the Mobile Server Host and Mobile Development Kit Host

To set up the Mobile Server host and the Mobile Development Kit (MDK) host, perform the following steps.

1. Install the Mobile Server and Mobile Development Kit.
2. Start the Mobile Server.
3. Create the requisite clients, publications, and subscriptions on the Mobile Server. Subsequent operations, such as Mobile Client installation and OLI command execution, are carried out on the MDK host. The MDK host contains a sample OLI configuration file named `oli.ini` and the OLI batch file named `oli.bat` at the following location:

   ORACLE_HOME`\Mobile\Sdk\bin`

9.3 Downloading the Client Binaries Into the Client Root Directory

To download the Mobile client binaries into the Mobile client root directory, the administrator performs the following:

1. Download `setup.exe` from the Mobile Server setup page, which is available at the following URL:

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

   Choose the appropriate Oracle Database Lite Mobile client. For example, choose Oracle Lite Branch Office for Branch Office applications, Oracle Lite WEB for Web applications, Oracle Lite WEB BC4J for Web applications that use BC4J, and the Oracle Lite WIN32 for WinCE (Windows Mobile) and Win32 applications.

   You use the same `setup.exe` that installs a client, but instead, install the client into a directory that will be used as a template for all of the clients you will be distributing.

2. Run the `setup.exe` using the distribution (`-d`) flag as follows.

   ```
   C:\CDSETUP\setup.exe -d
   ```

   Note: This program creates a `setup.exe` icon in the same folder to which you downloaded the executable. Running this without the `-d` flag, by double-clicking the icon, for instance, performs a normal setup of the client on your machine. The `-d` option must be used to create the necessary files for an offline install.

3. Enter your user name and password.

4. The **Choose Directory** screen prompts you for a destination directory where you want to create the client bundles. Ensure that you create a new destination directory other than the directory in which the `setup.exe` file resides. The appropriate files are copied to the destination directory.

   The install directory specified during this process is defined as `MOBILECLIENT_CD_ROOT` in the OLI configuration file (`oli.ini`). Once the client directory is established, the configuration file should be updated accordingly. All client databases will be instantiated in this directory. Hence, it is important to ensure adequate disk space availability.

   Note: Ensure that the `MOBILECLIENT_CD_ROOT` distribution directory is different from the `MOBILECLIENT_ROOT` directory.

   See Section 9.5, "Configure How OLI Creates the Client Distribution Packages With the OLI Configuration File" for information on the elements in the `oli.ini` file.

5. Finally, download the client binaries from the Mobile Server by successfully synchronizing the Mobile client with the Mobile Server. Using the `msync.exe` application (or either of the OC4J or Web-to-Go clients), initiate the synchronization process for one of the subscribing clients. It may be necessary to uninstall the existing Mobile client before running the Mobile client `setup.exe`. Synchronizing before offline instantiation accomplishes two things: the synchronization process validates the setup and downloads the client data (ODB files) that are associated with the subscription that should be included in the client bundle.

Note: This only downloads the binaries for the Mobile client. The user data is installed with the OLI process, as defined in Section 9.7, "Deploying Client Distribution Files on Client Machines".

9.4 The OLI Package Directory

The OLI package directory is a location where all the individual client packages are placed during the offline instantiation process. This directory must be located on a drive with adequate free disk space for all client databases. In the file `oli.ini`, this directory is called `OLI_CDS_ROOT`.

See Section 9.5, "Configure How OLI Creates the Client Distribution Packages With the OLI Configuration File" for information on the elements in the `oli.ini` file.

Note: Ensure that this directory is different from `MOBILECLIENT_CD_ROOT` and `MOBILECLIENT_ROOT`.

9.5 Configure How OLI Creates the Client Distribution Packages With the OLI Configuration File

The offline instantiation tool, OLI, reads the `oli.ini` file to determine how many users, the names of those users, the location of the Mobile client binaries, and so on. Before you use the offline instantiation tool, make sure that you set up these parameters correctly.

The offline instantiation tool and configuration file is located in the Mobile Development Kit, under *`ORACLE_HOME`*`\Mobile\Sdk\bin\`. Thus, make sure that you have installed the Mobile Development Kit.

The following describes how to configure the `OLI.INI` file:

- Section 9.5.1, "SETUP"
- Section 9.5.2, "USERS"
- Section 9.5.3, "Example of OLI.INI File"

9.5.1 SETUP

The `SETUP` section contains the general configuration for OLI to create the client packages.

MOBILE_SERVER

Provide the Mobile Server host and port that the Mobile clients will connect to for synchronization. You can supply a host name or an IP address. The default port number is 80. Sync server must be running when OLI is launched.

```
MOBILE_SERVER=myhost.us.oracle.com:80
```

USE_SSL

If you are going to use SSL, set to YES. Default is NO.

```
USE_SSL=NO
```

JDBC_URL

Configure the JDBC URL to the Mobile Server Repository.

```
JDBC_URL=jdbc:oracle:thin:@myhost.us.oracle.com:1521:oid
```

SCHEMA and PASSWORD

Configure the Mobile Server administration schema name and password for the Mobile Server Repository.

```
SCHEMA=MOBILEADMIN_SCHEMA
PASSWORD=MOBILEADMIN_PASSWORD
```

MAKEODB_METHOD

The `MAKEODB_METHOD` parameter defines how the client Oracle Lite databases are populated. The default and more performant option is `JDBC`, which transfers the data for all clients from the repository over a JDBC connection. Otherwise, you can configure `SYNC`, which uses the client-server synchronization for each individual client to generate the Oracle Lite databases.

```
MAKEODB_METHOD=JDBC
```

OLITE_JDBC_DRIVER

The `OLITE_JDBC_DRIVER` parameter defines the JDBC driver type for the connections to Olite client databases. Valid values are `NATIVE` and `ODBC`.

- `NATIVE`: The default; use the Oracle Database Lite native driver.
- `ODBC`: use the SUN JDBC-ODBC bridge.

```
OLITE_JDBC_DRIVER=NATIVE
```

MOBILECLIENT_ROOT

OLI needs to know where you installed the template Mobile client. This is the client that you installed in Section 9.3, "Downloading the Client Binaries Into the Client Root Directory". This Mobile client is for a user that is identical to all of the other users that will use this distribution. Thus, OLI uses this Mobile client as the template for how to build the distribution.

```
MOBILECLIENT_ROOT=C:\MOBILECLIENT
```

MOBILECLIENT_CD_ROOT

OLI needs to know where you installed the Mobile client binaries when you executed `setup -d`. Configure the directory that you specified in Section 9.3, "Downloading the Client Binaries Into the Client Root Directory". OLI packages these binaries with the same configuration as designated by the Mobile client installed in the `MOBILECLIENT_ROOT` directory for each user defined in the `USERS` section.

```
MOBILECLIENT_CD_ROOT=C:\MOBILECLIENT_CD
```

SHARED_CD_MODE

If set to YES, then only one generic client CD is generated and placed in the `<OLI_CDS_ROOT>/Shared_CD`. This CD only contains shared data. If set to NO, then each user has its own package created under the `<OLI_CDS_ROOT>/<username>`.

OLI_CDS_ROOT

This is the final directory where OLI places all of the client distribution packages for each user. From this directory, you can distribute the packages to each user.

```
OLI_CDS_ROOT=C:\OLI_CDS
```

DEVICE_TYPE

This specifies the type of device to which the client distribution packages are installed. You cannot install the packages to multiple platforms. Instead, choose one of the following:

- `WIN32`: Windows 32
- `WTG`: Web-to-Go client, Branch Office
- `WCE`: Windows CE (PocketPC)

```
DEVICE_TYPE=WIN32
```

THREADS

You can specify the number of threads OLI can use to process all of the users listed in the `OLI.INI` file. The more threads you allow, the more users can be processed concurrently.

9.5.2 USERS

The `USERS` section defines the users and their passwords. For each user, OLI creates a client distribution package that contains the Mobile client binaries. The clients must have been created on the Mobile Server, as described in Section 9.1, "Using Offline Instantiation to Distribute Multiple Mobile Clients". On each line, put the username and password, as follows:

```
[USERS]

CONSC1 MANAGER
CONSC2 MANAGER
```

For each user, a client distribution package is created.

9.5.3 Example of OLI.INI File

The following sample configuration file is available on the MDK host at *ORACLE_HOME*`\Mobile\Sdk\bin\`.

```
#########################################################################
#
# OLI.INI
# Oracle 10g Lite Offline Instantiation Configuration File
# Copyright © 1997-2004 Oracle Corporation.
# All Rights Reserved.
#
#########################################################################
#
# There are two sections whose names are enclosed in square
# brackets: [SETUP] and [CLIENTS].
# Lines starting with a "#", ";", "--" or "//" are comments.
#

#
# Site specific parameters.
# The format for this section is <PARAMETER> = <VALUE>
#
[SETUP]
```

```
#
# The mobile server name or IP. If on a port other than 80, append ":<port>".
# Sync server need be running when OLI is launched.
#
MOBILE_SERVER=hostname.domain:80

#
# If the mobile server port specified above is secure, set "USE_SSL" to "YES".
# Otherwise, use "NO".
#
USE_SSL=NO

#
# The mobile server database repository JDBC URL, mobileadmin schema and password
#
JDBC_URL=jdbc:oracle:thin:@hostname.domain:1521:oid
SCHEMA=MOBILEADMIN_SCHEMA
PASSWORD=MOBILEADMIN_PASSWORD

#
# The method used to populate client databases. Valid values are "SYNC" and
"JDBC".
# "SYNC": use client-server synchronization to generate ODBs.
# "JDBC": use JDBC to transfer data from server repository to client.
# If clients subscribe to same data for some tables, "JDBC" is faster since they
are
# transferred only once for all clients.
#
MAKEODB_METHOD=JDBC

#
# The JDBC driver type for the connections to Olite client databases. Valid values
are "NATIVE" and "ODBC".
# "NATIVE": use Olite native driver.
# "ODBC": use SUN JDBC-ODBC bridge.
#
OLITE_JDBC_DRIVER=NATIVE


#
# The destination directory for the Olite Mobile client installation.
# This is the directory when you run "setup.exe" without any options.
#
MOBILECLIENT_ROOT=C:\MOBILECLIENT

#
# The directory of the Olite Mobile client installation package.
# This is the destination directory when you run "setup.exe" with an option "-d".
#
MOBILECLIENT_CD_ROOT=C:\MOBILECLIENT_CD

# If set to YES only one generic client CD is generated and place in
# <OLI_CDS_ROOT>/Shared_CD. This CD only contains shared data. If set to NO,
# each user has its own package created under <OLI_CDS_ROOT>/<USERNAME>.
#
SHARED_CD_MODE=NO
#
# The Directory where OLI puts the client instantiation packages.
# Under this directory, each instantiated client will have a sub directory
# which can be copied to a CD to be used for Mobile client installation
```

```
# on the client machine. Client ODBs are included.
#
OLI_CDS_ROOT=C:\OLI_CDS

#
# The device type of the targeted Mobile client machines.
# Use "WIN32" for win32 native,
# use "WTG" for webtogo client deployments and
# use "WCE" for Windows Mobile
#
DEVICE_TYPE=WIN32

#
# The number of clients to be processed concurrently
#
THREADS=1

#
# List of clients to be instantiated. The clients must have been created
# on the Mobile server.
# The format for this section is <CLIENTID> <PASSWORD>
# Passwords are required
#
[USERS]
CONSC1 MANAGER
CONSC2 MANAGER
CONSC3 MANAGER
CONSC4 MANAGER
CONSC5 MANAGER
CONSC6 MANAGER
CONSC7 MANAGER
CONSC8 MANAGER
CONSC9 MANAGER
CONSC10 MANAGER
CONSC11 MANAGER
CONSC12 MANAGER
CONSC13 MANAGER
CONSC14 MANAGER
CONSC15 MANAGER
CONSC16 MANAGER
CONSC17 MANAGER
CONSC18 MANAGER
CONSC19 MANAGER
CONSC20 MANAGER
```

9.6 Using the OLI Engine to Create and Package the Client Distribution File

The OLI engine reads the file `oli.ini` in the current directory for information related to configuration settings. Before launching the OLI engine, you must edit the `oli.ini` file, as described in Section 9.5, "Configure How OLI Creates the Client Distribution Packages With the OLI Configuration File". The OLI engine provides commands that enable you to create and populate the client database files, create packages for Mobile clients, and cleanup OLI tables. As a normal practice, execute them in the given order.

Note: Before launching the OLI engine, you must do the following:

- Ensure that the JDK version is 1.4 or later.
- Ensure that the mobile devices for each user that will be a part of the OLI are NOT disabled.
- Edit the `oli.ini` file. The OLI engine uses two repository tables that store information related to resuming OLI tasks during interruptions or failures. These tables are `C$OLI_CLIENTS` and `C$OLI_SETUP`.

See Section 9.5, "Configure How OLI Creates the Client Distribution Packages With the OLI Configuration File" for information on the elements in the `oli.ini` file.

The OLI engine relies on a few Java classes and native libraries. To make the Java libraries and native libraries accessible to the OLI engine, the software contains a batch file named `oli.bat`, in which the necessary environment variables are set. Using the `oli.bat` file is recommended instead of directly using the Java class used by OLI, `oracle.lite.sync.OLI_Win32`.

To launch the OLI engine using the Command Prompt window, locate the directory *ORACLE_HOME*`\Mobile\Sdk\bin` and execute the `oli.bat` file at the Command Line.

Note: Shut down the OC4J or Web-to-Go client prior to executing the `oli.bat` file.

This action displays the following usage information. NOTE: You execute only ONE of the following commands at a time: `makeodb`, `package`, `cleanup`, or `check status`. Do NOT execute `oli.bat` with more than one of these commands. You will notice that the instructions show how to create the offline instantiation packages by executing `oli.bat` several times—once for each command.

```
Usage
-----
oli.bat [-d] [-i] [-g] [makeodb] [package] [cleanup] [check status]
-g  Debug mode
```

The command options for `oli.bat` are as follows:

- `-d`: If your `oli.ini` configuration file (see Section 9.5, "Configure How OLI Creates the Client Distribution Packages With the OLI Configuration File") is not in the default directory, specify where it is located with this flag.
- `-i`: If you renamed the `oli.ini` file to another name, specify the name with this option.
- `-g`: Turn on debugging.

To carry out OLI tasks, re-execute the command using the appropriate switches and arguments.

To build the client installation package, perform the following:

1. Section 9.6.1, "Create and Populate Client Database Files with the MAKEODB Command"

2. Section 9.6.2, "Package the Mobile Client Binaries with the Client Database Files with the package Command"
3. Section 9.6.3, "Clean Up the OLI Tables Before Executing OLI for Another Distribution"
4. Section 9.6.4, "Check the Status of OLI Clients"

Note: Before executing the `makeodb` and `package` commands on WinCE or Win32 devices, ensure that you set the `DEVICE_TYPE` parameter to `WCE`, `WTG`, or `Win32` in the `oli.ini` file.

9.6.1 Create and Populate Client Database Files with the MAKEODB Command

Creates and populates the client Oracle Lite database `.odb` files. For each user defined in the `[USERS]` section of the `oli.ini` file, OLI creates the client database files for subscribed publications.

Usage

```
oli.bat [-g] makeodb
```

The initial status of a client is `RESET`. After the first client is processed successfully by `makeodb`, then its status changes from `RESET` to `SLUG`. After all of the other clients are replicated using the first client, their status changes from `RESET` to `ODBMADE`. Finally, when the OLI engine packages the client information into a directory, the status changes to `PACKAGE`.

You can see the state of the client distribution files by executing the `checkstatus` command (see Section 9.6.4, "Check the Status of OLI Clients").

9.6.2 Package the Mobile Client Binaries with the Client Database Files with the package Command

After creating the client database file, execute the `package` command, which packages up the client database file and the Mobile client binaries for each user defined in the `[USERS]` section in the `oli.ini` file. Each client package is written to a subdirectory of the client name under the directory defined in the `OLI_CDS_ROOT` section in the `oli.ini` file.

Usage

```
oli.bat [-g] package
```

After a client package is successfully processed, its status is changed to `PACKAGED`. You can see the state of the client distribution files by executing the `checkstatus` command (see Section 9.6.4, "Check the Status of OLI Clients").

9.6.3 Clean Up the OLI Tables Before Executing OLI for Another Distribution

The `cleanup` command cleans the OLI tables. The `cleanup` command re-creates the OLI tables. Execute this command before executing OLI for another distribution. Do not execute this command without a need to start again.

Usage

```
oli.bat [-g] cleanup
```

9.6.4 Check the Status of OLI Clients

Check the status of OLI clients with the `checkstatus` command.

Usage

```
oli.bat checkstatus
```

9.7 Deploying Client Distribution Files on Client Machines

Once you have the client packages ready, you can distribute them to your users either by putting the distribution files on a CD for them or by giving the user access to the distribution files over the network or through email. Whether you use the CD or provide your users access to the distribution directory, the client must have network access to the Mobile Server. When using OLI to register the client, the connection is used to propagate the initial synchronization of data.

When you finish packaging the users using the OLI command, a directory is created in the `OLI_CDS_ROOT` directory for each user. In each subdirectory, the distribution files, with a `setup.exe`, is written. The user can execute the `setup.exe` directly from this subdirectory over a network, you can zip up all of these files and send the ZIP file to the user over email, or you can copy all of the files to a CD for each user. Once the user has access to the distribution files, the user executes the `setup.exe` to install the Mobile client binaries.

The deployment process for WinCE clients are different from those of native Win32 clients and Web-to-Go clients. The following sections describe how the user would install the distribution files on these devices:

- Section 9.7.1, "Deploy Win32 Native or Web-to-Go Client Distribution Package"
- Section 9.7.2, "Deploy WinCE PocketPC Client Distribution Package"

9.7.1 Deploy Win32 Native or Web-to-Go Client Distribution Package

To deploy on client devices for native Win32 platform or Web-to-Go clients, perform the following steps.

1. After a successful server side `OFF_LINE INSTANTIATION` process, each client is provided with a one-click installable package in the directory specified by the parameter named `OLI_CDS_ROOT` in the `oli.ini` file. The client sub-directory (package) is named after the client name. Provide each user with the client distribution package; for example, copy the client package to the client machine.
2. On the client device, perform the following:
   a. If you have a Mobile client installed, uninstall the existing software.
   b. From the client distribution package, run `setup.exe`.

9.7.2 Deploy WinCE PocketPC Client Distribution Package

To deploy on PocketPC client devices for WinCE clients, perform the following steps.

1. Install the Mobile client for Windows CE onto the CE device.
2. After a successful server-side `OFF_LINE INSTANTIATION` process, each client contains a package in the directory, which is specified by the parameter `OLI_CDS_ROOT` in the `oli.ini` file. The client sub-directory (package) is named after the client name. Provide each user with the client distribution package. Copy the client package to the *`ORACLE_HOME`* directory of the WinCE device.

3. Perform a synchronization.

9.8 Creating a Single Package or Shared CD for Users That Share Data

OLI enables an administrator to prepare a CD for each user, which contains the Mobile client binaries and the user's data in ODB files. This process creates one package, or one CD, for each user and may have the following drawbacks:

1. When there are a large number of users, the OLI process may be slow, resource intensive and error-prone.
2. When most of the initial data is the same for all users and in read-only lookup tables, generating separate CDs is not necessary.
3. Once a CD is used, the CD cannot be re-used for re-installation in cases like Mobile client corruption. If you try to re-use a CD, then a complete refresh may be triggered.

In shared CD mode, a generic CD is produced that contains the Mobile client binaries and shared data. The shared CD can be used and re-used by all users. Users need to specify the appropriate username and password during installation. The first synchronization after the install will bring down user-specific data and new shared data.

Follow the same directions as if you were creating the multiple packages, with the following differences:

1. Set the Instance Parameter `MAGIC_CHECK` to `NONSHARED`. The `MAGIC_CHECK` parameter enables you to control the magic number checking of publication items. If magic check is enabled for a publication item and there is a mismatch between server and client magic numbers, the publication item receives a complete refresh.

   When you set this to `NONSHARED`, then the magic check is enabled only for non-shared publication items. See Section 5.5, "Configuring Data Synchronization For Farm or Single Mobile Server" on where the Instance parameters are modified in the Mobile Manager. See the full description of the `MAGIC_CHECK` parameter in Section A.7, "[CONSOLIDATOR]".
2. Configure only a single username in the list of users at the end of the `oli.ini` file. See Section 9.5, "Configure How OLI Creates the Client Distribution Packages With the OLI Configuration File" for an example.
3. Configure the `SHARED_CD_MODE` element in the `oli.ini` file to `YES`. See Section 9.5, "Configure How OLI Creates the Client Distribution Packages With the OLI Configuration File" for an example.
4. After you complete the packaging of the offline instantiation by executing the `oli.bat` package command (see Section 9.6, "Using the OLI Engine to Create and Package the Client Distribution File"), then copy the contents of the `<OLI_CDS_ROOT>/SHARED_CD` directory onto a CD.
5. Place the CD into the CD drive of the client device, change directory to the CD and execute the following:

   Note: The client must have network access to the Mobile Server. When using OLI to register the client, the connection is used to propagate the initial synchronization of data.

```
setup.exe setup.inf <username> <password>
```

6. After you have finished installing all of your clients and have performed the first synchronization for each of them, change the `MAGIC_CHECK` parameter in the Data Synchronization Instance parameter bakc to the default of `ALL`.

10

Using the Application Server OID With Mobile Server

If you decide to use OID in OracleAS, then after you install the application server and Oracle Mobile Lite, perform the following:

1. Migrate any existing Mobile users to OID (See Section 3.6, "Migrate Your Users From the Mobile Server Repository to the Oracle Internet Directory" in the *Oracle Database Lite Getting Started Guide*).
2. Use the Mobile Manager to edit the configuration file (`webtogo.ora`) and set `SSO_ENABLED=YES`. Do not edit the `webtogo.ora` file directly. In the Mobile Manager, migrate to the Administration tab and select **Edit Config file**. This is the `webtogo.ora` file.
3. Restart the application server.
4. Navigate to the Users tab in the Mobile Manager, which displays all users in OID. Select the users to enable for Mobile Server and enable these users. Assign the appropriate application to these users.
5. Install the Mobile client and synchronize using one of the users you enabled.

11

Configure Security in Oracle Database Lite

The following sections detail how to manage security in Oracle Database Lite:

Note: There is additional information about developing for security in the "Security" chapter in the *Oracle Database Lite Developer's Guide*.

11.1 Security Enhancements

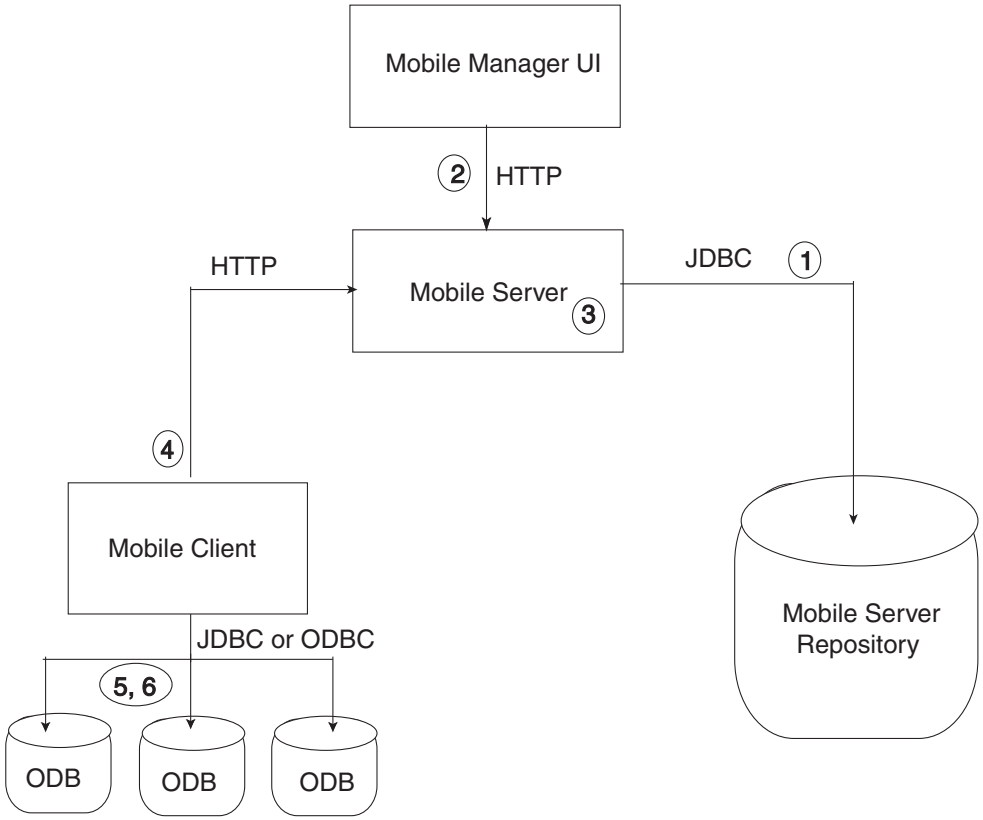
A number of security enhancements have been made, as follows:

- It is possible to restrict the database privileges for the `MOBILEADMIN` user, once the application has been published.
- Passwords for all default accounts can be chosen at install time.
- Remote HTTP access to the Mobile Client Web-to-Go has been disabled by default.
- The Windows Service for the Branch Office listener runs under a restricted Windows user. In fact, the Windows services run under a non-priviledged user in the Branch Office configuration.

11.2 Which Password is Which?

In the Oracle Database Lite product, there are several username/password combinations that are used for different security reasons and for separate types of users or administrators. This section describes each of the username/password combinations in order to eliminate confusion.

Figure 11–1 Components That Use Passwords



As shown in Figure 11–1, the passwords for the Mobile client environment are as follows:

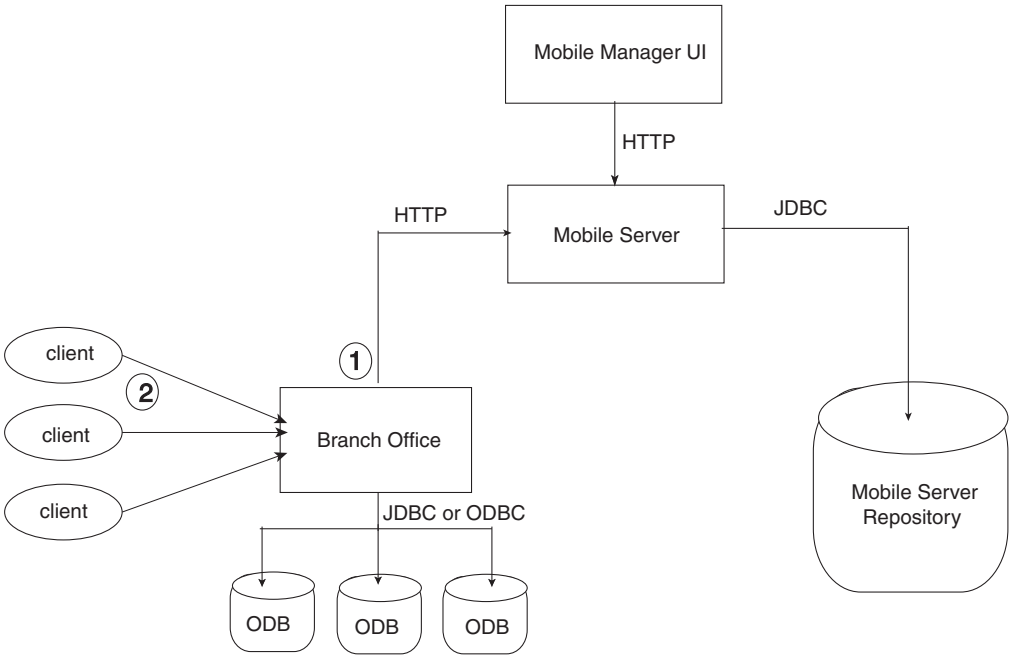
1. When the Mobile Server accesses the Mobile repository, it uses the repository username/password. This defaults to the user `MOBILEADMIN` and the password is set during install. When the user accesses the user data in the Mobile Server repository, the Mobile Server connects using the Mobile repository username, where the Mobile user name and password are authenticated before access is provided to the user data.
2. The Mobile Manager is a Web administration UI that provides administrators the ability to modify how the Mobile Server behaves. Only administrators can use this tool; thus, only Mobile Server administrators can log in with their passwords. The initial administration username/password is created during installation. For adding or modifying, use the Mobile Manager to modify these on the Mobile Server.
3. Within the Mobile Server, there are two types of username/password combinations—both of which are created using the Mobile Manager UI: administrator and Mobile user.
4. The Mobile user provides its username/password when synchronizing, which Mobile Server uses to verify that this use can access the application data. The Mobile username and password are stored in two places: the Mobile Server repository and the Oracle Lite database associated with this user (the ODB file). Thus, when you modify the Mobile user password, you must perform one of the following:
   - Modify the password on the client using either mSync UI or Client Workspace. Only modify the password using these tools 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.

5. Every Oracle Lite database (ODB file) has the `SYSTEM` user for connecting. Oracle Database Lite uses `SYSTEM` as the username and then the Mobile user password as the password when connecting during synchronization.
6. By default, each ODB file is not encrypted. If you want to encrypt the database, perform one of the following:
   - If you are using the Oracle Lite database as part of the Mobile option, where you are syncrhonizing the user data to a back-end server, then set the `polite.ini ENCRYPT_DB` parameter. Oracle Database Lite will then automatically encrypt the Oracle Lite database with the user password. For details on this parameter, see Section F.3.2.12, "ENCRYPT_DB".
   - If you are using the Oracle Lite database as an embedded database or standalone database—with no synchronization ability—then encrypt the database using the `encrypdb` utility and provide your own password to use for encrypting the database. See Appendix A.4 "ENCRYPDB" in the *Oracle Database Lite Developer's Guide* for details.

Figure 11–2 Branch Office Passwords



If you have a Branch Office configuration, then the following additional username/password combinations exist:

1. Branch Office is installed as a Windows server with the username of `OracleDatabaseLite` with the minimum set of privileges required to execute

the Oracle Database Lite software. Only the Branch Office can initialize the synchronzation; the remote clients cannot.

Normally, when installed, the password for the `OracleDatabaseLite` user is randomly generated during the setup. You can either pre-configure this password before the Branch Office installation or modify it after the configuration. See Section 3.5.3, "Defining Password for OracleDatabaseLite User for Branch Office on Windows Machine" in the *Oracle Database Lite Getting Started Guide*.

2. The remote clients access the Oracle Lite database through Branch Office; thus, the username/password for the remote client is validated by Branch Office.

11.3 Configuring SSL For Mobile Server

Oracle Database Lite supports Secure Socket Layer (SSL) communication between the Mobile Server and Mobile clients. Oracle Database Lite uses the SSL that is embedded within OC4J, which is shipped as part of Mobile Server.

Note: If you choose to install standalone Mobile Server, the standalone OC4J is installed; otherwise, Mobile Server is installed on top of an existing OracleAS stack. OracleAS also includes OC4J, but the configuration for SSL is more involved. This chapter covers the basic SSL configuration for the standalone Mobile Server. See the *Oracle Application Server Containers for J2EE Security Guide* for more information on all aspects of configuring SSL for OracleAS.

This chapter assumes that you understand the concepts behind SSL and provides only the steps for using keys and certificates for SSL communication for the standalone Mobile Server.

- Section 11.3.1, "Creating an SSL Certificate"
- Section 11.3.2, "Configuring Mobile Server for SSL"
- Section 11.3.3, "Enabling SSL Authentication for Web-to-Go Clients"
- Section 11.3.4, "Troubleshooting Error Messages for an SSL-Enabled Mobile Server"
- Section 11.3.5, "Client-Side Configuration for Secure Socket Layer (SSL)"

Note: These are server-level steps which are typically executed prior to deployment of an application that requires SSL communication.

11.3.1 Creating an SSL Certificate

SSL communicates by validating an SSL certificate between the client and the server. This section describes how to ceate the SSL certificate. However, often when you are first starting with new functionality, you may want to use a temporary certificate just to see how the SSL functionality works.

Oracle Database Lite ships a sample keystore file with a self-signed sample certificate. The password for this sample keystore file is `oracle`. Use this keystore only for development or testing purposes. Obtain a signature from a recognized certificate authority for all production systems. The test keystore is located in the following directory:

ORACLE_HOME`\Mobile\Server\Bin\samplekeystore`

To create a keystore file, perform the following steps:

1. Use the Sun Microsystems Java `keytool` utility to generate a private key, public key, and an unsigned certificate. Place this information into either a new or existing keystore.

   Note: A keystore is a `java.security.KeyStore` instance that you create and manipulate using the `keytool` utility, which is provided with the Sun Microsystems JDK. See `http://java.sun.com/j2se/1.5.0/docs/tooldocs` for more information on the `keytool` utility.

2. Obtain a signature for the certificate, using either of the following approaches:
   - Generate your own signature by using `keytool` to self-sign the certificate. This is appropriate only if your clients trust you as your own certificate authority.
   - Obtain a signature from a recognized certificate authority through the following steps:

   a. Using the certificate from Step 1, use `keytool` to generate a certificate request, which requests a certificate authority to sign the certificate.

   b. Submit the certificate request to a certificate authority.

   c. Receive the signature from the certificate authority and import it into the keystore using `keytool`. In the keystore, the signature is matched with the associated certificate.

If you install the Mobile client using `setup.exe` after you create the self-signed certificate, then a message pops up asking if you want to continue. If you click Yes, then a parameter is added to the `polite.ini` that tells Oracle Database Lite to not validate the certificate. However, if you install the Mobile client using any other method, you need to set this parameter yourself. Set the `SSL_IGNORE_CERT` parameter in the `polite.ini` file to 1.

Each certificate authority has its own process for requesting and receiving signatures. Since this is outside the scope and control of Oracle Database Lite, it is not covered in Oracle Database Lite documentation. However, the SSL section in the *Oracle Application Server Containers for J2EE Security Guide* has an example of how to generate your own keystore. For other information, go to the Web site of any certificate authority. Each browser lists trusted certificate authorities.

Here are the Web addresses for VeriSign, Inc. and Thawte, for example:

```
http://www.verisign.com/
http://www.thawte.com/
```

11.3.2 Configuring Mobile Server for SSL

Once you have a certificate, you must configure SSL in the application server that is installed with the Mobile Server. When you installed, you chose to install the Mobile Server either in standalone mode or to use the application server. Both of these environments are discussed below:

- Section 11.3.2.1, "Configuring SSL for Mobile Server With OracleAS"
- Section 11.3.2.2, "Configuring SSL for Standalone Mobile Server"

11.3.2.1 Configuring SSL for Mobile Server With OracleAS

For production systems, you install OracleAS before you install the Mobile Server. You must configure SSL on both the application server and the Mobile Server, as follows:

1. Configure SSL in the application server using the administration GUI. The directions on how to configure SSL when using OracleAS as your middle-tier is in the SSL or HTTPS chapter in the *Oracle Application Server Containers for J2EE Security Guide*.
2. Configure SSL in the Mobile Server by adding `SSL=YES` in the `[WEBTOGO]` section of the `webtogo.ora` file. In the Mobile Manager, migrate to the Administration tab and select **Edit Config file**. This is the `webtogo.ora` file.
3. After all configuration is complete, restart the application server to initialize the changes.

11.3.2.2 Configuring SSL for Standalone Mobile Server

With the standalone Mobile Server, the standalone version of the OC4J application server is installed with the Mobile Server. To configure SSL for this environment, you modify the Mobile Server `webtogo.ora` file and certain XML elements within the OC4J XML configuration files, as follows:

1. Configure SSL in the Mobile Server by adding `SSL=YES` in the `[WEBTOGO]` section of the `webtogo.ora` file. In the Mobile Manager, migrate to the Administration tab and select **Edit Config file**. This is the `webtogo.ora` file.
2. If you do not have a `secure-web-site.xml` file, then copy and rename the `default-web-site.xml` to *`ORACLE_HOME`*`\mobile_oc4j\j2ee\mobileserver\config\secure-web-site.xml`.
3. Edit the `secure-web-site.xml` file with the following elements:
   a. Add `secure="true"` to the `<web-site>` element, as follows:

      ```
      <web-site port="443" display-name="Oracle Application Server Containers for
      J2EE Web Site" secure="true">
      ```

   b. Add the following new line inside the `<web-site>` element to define the keystore and the password:

      ```
      <ssl-config keystore="YourKeystore" keystore-password="YourPassword" />
      ```

      where *`YourKeystore`* is the path and name of the keystore and *`YourPassword`* is the keystore password. The path for the keystore can either be a full path or a path that is relative to *`ORACLE_HOME`*`\j2ee\mobileserver\config`. In addition, you can hide the password through password indirection. This is discussed fully in the *Oracle Application Server Containers for J2EE Security Guide*. For example, with a keystore of "`../../keystore`" and password of "`oracle`", the configuration is as follows:

      ```
      <!-- Enable SSL -->
      <ssl-config keystore="..\..\..\mobile\server\bin\samplekeystore"
                  keystore-password="oracle"/>
      ```

   c. Change the `<web-site>` element port number to use an available port. The reason you must change the port is because you copied this file from `default-web-site.xml`, which uses the port that is currently configured. Thus, choose a port that can be used for SSL communication; for example, the default for SSL ports is 443.

d. Save the changes to `secure-web-site.xml`.

4. Edit the `server.xml` file to point to the `secure-web-site.xml` file.

   a. Uncomment or add the following line in the file `server.xml` so that the `secure-web-site.xml` file is added to the OC4J initialization.

   ```
   <web-site path="./secure-web-site.xml" />
   ```

   b. Save the changes to the `server.xml` file.

5. Stop and re-start OC4J to include the `secure-web-site.xml` file modifications.

6. Test the SSL port by accessing the Mobile Server in a browser on the SSL port. For example, `https://<yourserver>:443/webtogo`.

   If you are using the test keystore file or your own self-signed certificate, you will be asked to accept the certificate, since the SSL certificate used is not signed by an accepted certificate authority. When completed, Mobile Server listens for SSL requests on the port configured in the `secure-web-site.xml` file and listens for non-SSL requests on the port configured in the `default-web-site.xml` file. You can disable either SSL requests or non-SSL requests, by commenting out the appropriate `*web-site.xml` in the `server.xml` configuration file.

   ```
   <web-site path="./secure-web-site.xml" /> - comment out this to remove SSL
   <default-site path="./default-web-site.xml" /> - comment out this to remove
   non-SSL
   ```

11.3.3 Enabling SSL Authentication for Web-to-Go Clients

For most Mobile clients, the SSL libraries used are the Microsoft SSL libraries. In this case, the SSL handshake occurs seamlessly. However, Web-to-Go clients use Oracle SSL libraries, which assumes that the SSL certificate is installed on the Mobile client.

In order to have the SSL certificate automatically installed on the Web-to-Go Mobile client, perform the following:

1. Upload the SSL certificate, which you created to the Mobile Server, as follows:

   a. Navigate to the Administration page for your Mobile Server in the Mobile Manager.

   b. Click **Server Certificate**.

   c. As shown in Figure 11–3, browse to the SSL certificate and click **Upload**.

Figure 11–3 Upload SSL Certificate

2. Download the `setup.exe` for the Web-to-Go Mobile client. The SSL certificate that was uploaded is automatically installed with the Web-to-Go Mobile client.

11.3.4 Troubleshooting Error Messages for an SSL-Enabled Mobile Server

The following errors may occur when using SSL certificates on your Mobile Server:

No available certificate corresponds to the SSL cipher suites which are enabled

Cause: Something is wrong with your certificate.

Action: Examine your certificates and check that at least one of them supports the SSL cipher suite you are using.

IllegalArgumentException: Mixing secure and non-secure sites on the same ip + port

Cause: You cannot configure SSL and non-SSL Web sites to listen on the same port and IP address.

Action: Check to see that different ports are assigned within `secure-web-site.xml` and either `default-web-site.xml` or `http-web-site.xml` files.

11.3.5 Client-Side Configuration for Secure Socket Layer (SSL)

As the end user, you can configure the Mobile client for OC4J or Web-to-Go to establish an SSL connection between the Mobile client and the Mobile Server.

The following sections describe how to enable SSL for your Mobile client:

- Section 11.3.5.1, "Communication between the Mobile Client and the Mobile Server"
- Section 11.3.5.2, "Connection between the Browser and the Mobile Client for OC4J or Web-to-Go"
- Section 11.3.5.3, "Support for Non-SSL Mobile Clients"

11.3.5.1 Communication between the Mobile Client and the Mobile Server

Based on whether or not you download the Mobile client for OC4J or Web-to-Go from the Mobile Server running in SSL, you can choose to configure communication between the Mobile client for OC4J or Web-to-Go and the Mobile Server. The following sections provide a description of configuring communication between the Mobile client and the Mobile Server.

- Section 11.3.5.1.1, "Mobile Client Download from a Mobile Server which is Running in SSL Mode"
- Section 11.3.5.1.2, "Mobile Client Download from a Mobile Server which is not Running in SSL Mode"

11.3.5.1.1 Mobile Client Download from a Mobile Server which is Running in SSL Mode The Mobile client for OC4J or Web-to-Go which is downloaded from the following URL is automatically configured for SSL and does not require manual configuration on the part of the end user. This download enables the Mobile client to communicate with the Mobile Server in SSL mode.

```
https://<mobile_server>:<port>/setup
```

11.3.5.1.2 Mobile Client Download from a Mobile Server which is not Running in SSL Mode If you have downloaded the Mobile client for OC4J or Web-to-Go from a Mobile Server, which is not running in SSL mode, modify the `SERVER_URL` parameter in the `webtogo.ora` file as follows.

```
SERVER_URL=https://<mobile_server>:<port>/webtogo/setup
```

Note: in the location bar, you must type `https`, to specify and indicate the SSL Mode, and not `http`.

11.3.5.2 Connection between the Browser and the Mobile Client for OC4J or Web-to-Go

While trying to connect to the Mobile client for OC4J or Web-to-Go in SSL mode, you will not be able to connect to the Mobile client, even if the following conditions exist.

1. The Mobile Server is running in SSL mode, as a module of Oracle9*i*AS.
2. The Mobile client for OC4J or Web-to-Go is also running in SSL mode.

To connect to the Mobile client for OC4J or Web-to-Go using a browser, you must specify `HTTP` and not `HTTPS` in the client URL, although the communication between the client and the server is through the `HTTPS` protocol.

For example, `http://<client_machine>/webtogo`

11.3.5.3 Support for Non-SSL Mobile Clients

Mobile Servers running in SSL mode possess the ability to synchronize with any Mobile client which is running in SSL or non-SSL mode. But, in the case of the Mobile client for OC4J or Web-to-Go, the client also runs in SSL mode to synchronize with the Mobile Server, which is running in SSL mode.

As SSL is not supported on many Mobile clients, the Mobile Server, which, is running in SSL mode, still supports Mobile clients that are running in non-SSL mode.

Note: Inside the Oracle Application Server, the Mobile Server runs on both SSL and non-SSL ports, to support SSL and non-SSL clients. The application server must be configured to run on both SSL and non-SSL ports, as a default function.

11.4 Encrypting the Client Oracle Lite Database

Whether you are using a Mobile client or an embedded application, you can encrypt the database used on the client.

- When encrypting the Oracle Lite database used with the Mobile client, modify the `POLITE.INI` file to set the `ENCRYPT_DB` parameter. See Section F.3.2.12, "ENCRYPT_DB" for more information.
- When encrypting the Oracle Lite database that is used with an embedded application—which does not use synchronization—use the `encryptdb` executable, as described in Appendix A.4 "ENCRYPDB" in the *Oracle Database Lite Developer's Guide*.

11.5 Using a Firewall Proxy or Reverse Proxy

Normally, the Mobile client synchronizes data inside a firewall on the corporate intranet, where the Mobile Server also resides. However, what if the user wishes to synchronize the Mobile client either from the internet, which is outside the firewall to a Mobile Server that exists inside the firewall? Or what if the Mobile Server exists on the public internet and the Mobile client is inside the firewall on the corporate intranet? Either way, you have to modify your configuration to enable a Mobile client and Mobile Server to communicate through a firewall.

One can think of many scenarios in which case mobile users want to connect to the corporate server, without being directly connected to the corporate intranet. The corporate firewall uses proxy support to allow the mobile user to connect to the server.

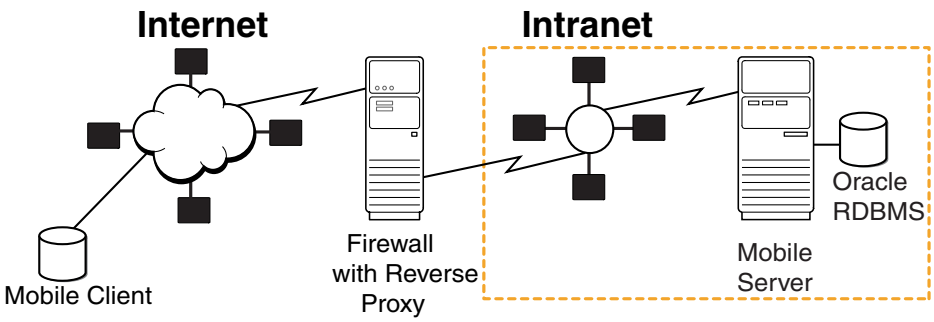
The following sections describe how to configure the Mobile Server and Mobile client to communicate through a firewall:

- Section 11.5.1, "Using Reverse Proxy to Communicate from Internet to Intranet"
- Section 11.5.2, "Using HTTP Proxy to Communicate From Inside a Firewall"

11.5.1 Using Reverse Proxy to Communicate from Internet to Intranet

If you are traveling to a customer site and you want to synchronize over the internet to the Mobile Server inside your corporate firewall, use a reverse proxy to communicate. A reverse proxy is used whenever a client outside a corporate network wants to connect to a resource available inside the corporate network, as shown in Figure 11–4. The corporate network is protected by a firewall, which stops the outside world from having direct access with the systems inside the corporate network. However, the reverse proxy enables designated traffic that originates outside the corporate network to reach servers inside the corporate intranet.

Figure 11–4 Mobile Client Communicating With Mobile Server Through Firewall Using Reverse Proxy



When you configure the reverse proxy, then the Mobile client communicates directly with the reverse proxy, which turns around and communicates with the Mobile Server.

> **Note:** The authentication on a reverse proxy server is supported only if the Mobile client's username and password are identical to those on the proxy.

In order for this communication to occur seamlessly, do the following:

- Section 11.5.1.1, "Configure the Apache Web Server as a Reverse Proxy"
- Section 11.5.1.2, "Set Up Mobile Server for Mobile Client Download"
- Section 11.5.1.3, "Download Reverse Proxy Mobile Client"
- Section 11.5.1.4, "Enable SSL When Using a Reverse Proxy"
- Section 11.5.1.5, "Configure Device Management to Work With a Firewall"

11.5.1.1 Configure the Apache Web Server as a Reverse Proxy

You need to set up the Apache Web Server software for the reverse proxy, as follows:

1. First, use Apache 2.0 or later for your proxy.
2. Configure the proxy server to point to the Mobile Server. See the Apache Web Server documentation for instructions on how to do so.

3. Set the following parameter in the `httpd.conf` configuration file:

```
BrowserMatch MSIE AuthDigestEnableQueryStringHack=On
```

4. When you use reverse proxy authentication, you must upper-case the username of the proxy digest.

5. If you are using authentication, then configure the Reverse Proxy with the username/passwords for all Mobile clients that will access this reverse proxy for synchronization.

   When the msync is launched, the username/password is sent automatically to the reverse proxy for authentication; thus, if the reverse proxy is not configured with the username/password, then the connection is refused.

11.5.1.2 Set Up Mobile Server for Mobile Client Download

If you know that the Mobile client is going to be accessing the Mobile Server through a reverse proxy, then you need to configure Mobile Server with the proxy server URL. This ensures that when the `setup.exe` is downloaded by the client, that the client is automatically configured with the reverse proxy URL, instead of the Mobile Server URL.

So, before you download `setup.exe` to the Mobile client, perform the following on the Mobile Server:

1. If your server is a Windows XP machine, you must have the Service Pack 2 installed.

2. Configure the Mobile Server to accept communication from the reverse proxy.

   Configure the `reverse_proxy` parameter in the `webtogo.ora` configuration file on the Mobile Server, as follows:

```
[WEBTOGO]
REVERSE_PROXY=http://<reverse_proxy_hostname>:<port_number>/webtogo
```

11.5.1.3 Download Reverse Proxy Mobile Client

After you have updated the Mobile Server with the proper reverse proxy configuration, perform the following on the client:

1. Configure the Mobile client to communicate with the reverse proxy in one of the two following methods:

   - If you configured the Mobile Server as described in Section 11.5.1.2, "Set Up Mobile Server for Mobile Client Download", then you can download the Mobile client software directly from the setup UI. The configuration automatically points to the reverse proxy when you perform the installation of the Mobile client.

   - However, if you installed the Mobile client software from within the corporate intranet or you have a client already installed on a machine, then you must modify its configuration. Modify the `polite.ini` configuration file for your Mobile client. Change or add the `SERVER_URL` parameter in the `NETWORK` section of the `polite.ini` configuration file to point to the host/port of the reverse proxy server, as follows:

     ```
     SERVER_URL=HTTP://<reverse_proxy_host>:<port>/webtogo
     ```

     If you use the `msync.exe` to synchronize, then enter the hostname of the reverse proxy in the Server box.

> **Note:** If you are planning on using the Mobile client both inside and outside of the corporate internet, you may want to have two `SERVER_URL` definitions—one for the internal corporate Mobile Server address and one for the reverse proxy address. Then, comment the one that you are not using and uncomment the one that you are using.

2. Perform post-installation steps for the Mobile client:

   If the Mobile client is a Windows client—such as Windows XP/2000 and WinCE devices—then Oracle Database Lite uses the WININET API for SSL over HTTP.

   The following are known issues when using SSL over HTTP:

   - The HTTP connection may slow down if you have the `Auto Detect Proxy` enabled in the Internet Explorer. In addition, it may also slow down if you do not have a proxy server in your network. In this case, uncheck the `Automatically detect proxy` option in the Internet Explorer.
   - For Windows 2000 clients, mSync may hang if you do not have all of the Microsoft patches applied.
   - If your Mobile Server or Reverse Proxy does not have a valid SSL certificate, then the Oracle Database Lite clients may stop working. This is critical if there are errors in Certificate chaining. See Section 11.5.1.4, "Enable SSL When Using a Reverse Proxy" for details.

11.5.1.4 Enable SSL When Using a Reverse Proxy

Normally, when you have a browser and you specify HTTPS for the connection between the browser and a reverse proxy, then the browser prompts for a username/password for authentication. However, with msync, a browser is not displayed. Instead, msync sends on the username/password for the user to the reverse proxy. Thus, you must have your environment configured correctly or the connection fails.

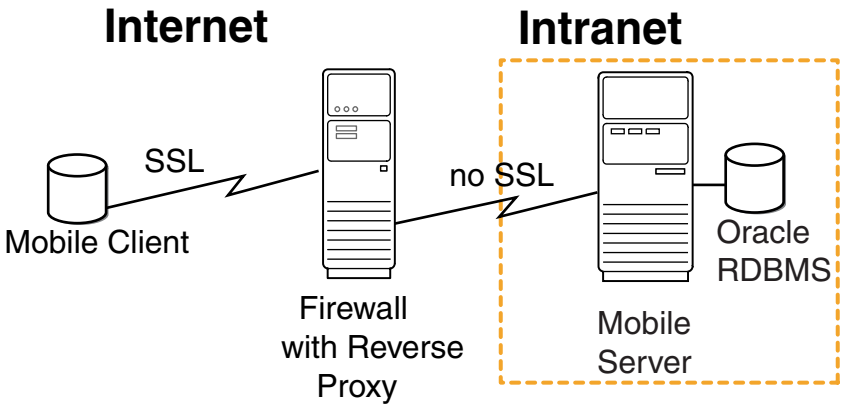
The following describes several scenarios that you may have between the Mobile client and the reverse proxy:

- Section 11.5.1.4.1, "Using SSL Authentication"
- Section 11.5.1.4.2, "Using SSL Between Mobile Client and Reverse Proxy"
- Section 11.5.1.4.3, "Using SSL Between Firewall and Mobile Server"
- Section 11.5.1.4.4, "Using Certificates That Are Not Signed By Trusted Authority"

11.5.1.4.1 Using SSL Authentication When you are using a reverse proxy firewall, SSL client authentication is not supported. You can only turn on server-side HTTPS authentication.

11.5.1.4.2 Using SSL Between Mobile Client and Reverse Proxy As Figure 11–5 demonstrates, you may want to encrypt your data and authenticate using SSL when using a reverse proxy.

Figure 11–5 Mobile Client Communicating Over SSL Through Firewall Using Reverse Proxy

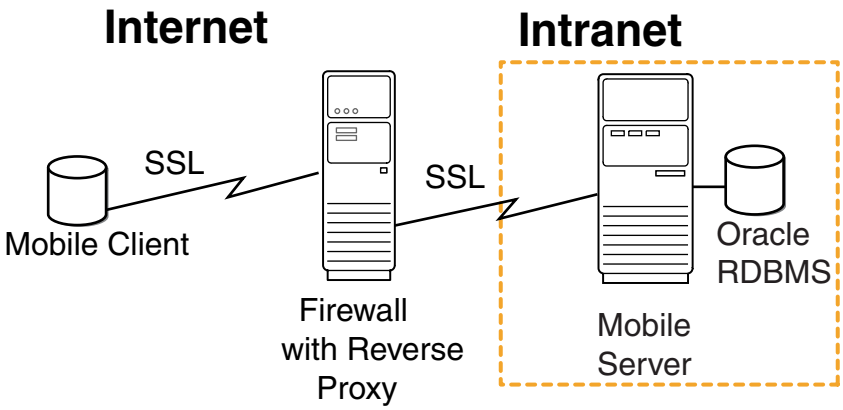


In this case, you must install the SSL certificate on the firewall for the SSL handshake between the Mobile client and the firewall. However, for the Web-to-Go client, you must upload the same certificate on the Mobile Server as described in step 4 in Section 11.3.3, "Enabling SSL Authentication for Web-to-Go Clients".

If you are using a certificate that is not signed by a trusted authority or if you want to disable SSL authentication, see Section 11.5.1.4.4, "Using Certificates That Are Not Signed By Trusted Authority".

11.5.1.4.3 Using SSL Between Firewall and Mobile Server As Figure 11–6 demonstrates, you may want to encrypt your data and authenticate using SSL when using a reverse proxy for all communication between the Mobile client and the Mobile Server. In this case, you would configure SSL between the Mobile client and the firewall; as well as configure SSL between the firewall and the Mobile Server.

Figure 11–6 Mobile Client Communicating Over SSL Through Firewall Using Reverse Proxy



In this case, you need to create a chained certificates with the following two certificates:

- A certificate for the connection between the Mobile client and the reverse proxy firewall.
- A certificate for the connection between the reverse proxy firewall and the Mobile Server.

Perform the following:

1. Create a chained SSL certificate that contains both the certificates from the reverse proxy followed by the certificate for the Mobile Server.
2. Install this certificate on the reverse proxy firewall for the Mobile client handshake.
3. If you are using a Web-to-Go client, install the chained certificate in the Mobile Server, as described in Section 11.3.3, "Enabling SSL Authentication for Web-to-Go Clients".

In general, install the chained certificate on the firewall for the SSL handshake between the Mobile client and the firewall. However, for the Web-to-Go client, you must upload the same certificate on the Mobile Server as described in step 4 in Section 11.3.3, "Enabling SSL Authentication for Web-to-Go Clients".

11.5.1.4.4 Using Certificates That Are Not Signed By Trusted Authority You can use certificates that are not signed by a trusted authority on the Mobile Server. A Web-to-Go client will use any certificate for encryption without any configuration modifications. However, for all other clients, if you are using a certificate that is not signed by a trusted authority, such as a self-signed certificate, then set the following parameter in the `NETWORK` section in the `polite.ini` (`polite.txt`) file on the client device:

```
DISABLE_SSL_CHECK=YES
```

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

11.5.1.5 Configure Device Management to Work With a Firewall

We use device management to send commands to devices for updates, initiating synchronization, as well as other commands. Device management uses HTTP as its communication protocol. So, if a firewall is in between the device and the Mobile Server, you must perform some configuration to enable device management communication.

There are two types of device management requests:

- Device initiated: The device management agent (dmagent), which is included on the Mobile device, registers with the Mobile server at device bootstrap. This is known as HTTP PULL, since the Mobile device polls the Mobile Server for any outstanding commands. The dmagent periodically polls the Mobile server for command requests on the Mobile Server listening port.
- Mobile Server initiated: This is known as HTTP PUSH, since the Mobile Server sends the commands directly to the Mobile device. You can send commands to one or more devices through the Mobile Manager or Java APIs. However, this is unusual, since most communication/synchronization is initiated from the client. Thus, the proxy must be configured correctly to enable communication initiated from the Mobile Server.

The following describes how to configure your Mobile Server to enable device management requests to a Mobile Server that resides behind a firewall:

- Section 11.5.1.5.1, "Configure Mobile Device Listening Port"
- Section 11.5.1.5.2, "Firewall Configuration"
- Section 11.5.1.5.3, "Proxy Configuration for the Mobile Server"

11.5.1.5.1 Configure Mobile Device Listening Port In the Mobile Server initiated scenario, the Mobile device has a listener with which the Mobile Server connects. Thus, the Mobile Server communicates with the dmagent listening port. The dmagent on the Mobile device, by default, listens on port 8521, which is configured in the `PUSH_PORT` parameter.

For all future client installations, you may modify the `PUSH_PORT` for all the clients by using Mobile Manager in the Mobile Devices -> Administration -> Configuration Management page. This change affects only the client installations that are performed after the modification.

11.5.1.5.2 Firewall Configuration Your firewall should be configured so that HTTP traffic is enabled in the following manner:

- The dmagent on the Mobile device should be able to access the `SERVER_PORT` on the firewall.
- The Mobile Server should be able to access the `PUSH_PORT` of all devices.

11.5.1.5.3 Proxy Configuration for the Mobile Server If the Mobile Server is behind the firewall, it can only access Mobile devices through a proxy. To configure the proxy server information in the metadata of the HTTP provider, navigate to Mobile Devices -> Administration -> Network Management -> HTTP in the Mobile Manager.

The metadata is any user-defined string that is required by the Java class during initialization. The HTTP provider metadata is a sequence of name-value pairs, where the name and value are separated by and equals sign ('='). Each pair is separated by the ampersand ('&'). This setting is effective when you send commands from a standalone Java program using device management APIs.

The following example adds the `PROXY` name-value pair with the proxy URL into the metadata after the `TIMEOUT` name-value pair:

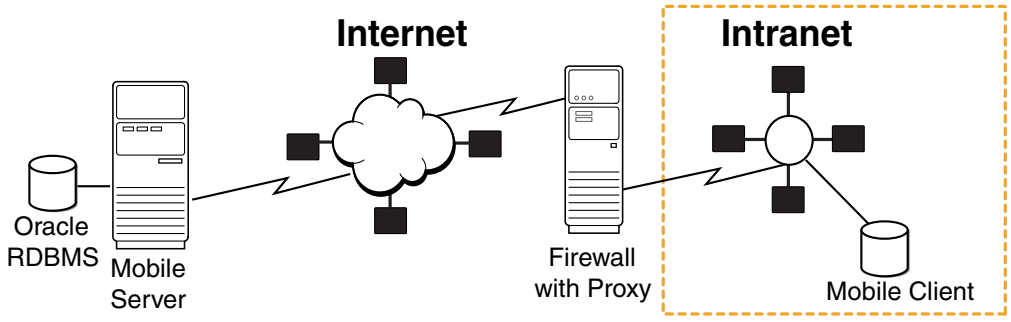
```
TIMEOUT=30&PROXY=http://proxy.foo.com:8080
```

11.5.2 Using HTTP Proxy to Communicate From Inside a Firewall

Use the HTTP proxy for clients inside a corporate network that want to connect to a resource on the Internet. As shown in Figure 11–7, the corporate network is protected by a firewall, which blocks direct access from inside the corporate network to the outside world. However, you can configure a proxy server on the firewall to allow designated traffic travel through the firewall.

As demonstrated by Figure 11–7, A mobile user may wish to use a public Internet connection to connect to the corporate network, using one of the many available wireless 802.11 hotspots.

Figure 11–7 Client Accessing Mobile Server on Internet



If the Mobile client is located in the corporate intranet and the Mobile Server is located somewhere in the public Internet—where both are separated by a firewall—then the firewall must be configured to let HTTP traffic travel through by means of a proxy server.

To enable communication from the Mobile client to a Mobile Server outside the corporate firewall, perform one of the following:

> **Note:** You may be able to set up the proxy for communication originated from the client in this scenario; however, we do not support server-initiated device management requests in this scenario.

- Section 11.5.2.1, "Proxy Configuration for Web-to-Go Clients"
- Section 11.5.2.2, "Proxy Configuration for All Other Clients"
- Section 11.5.2.3, "Proxy Configuration for the Device Management Agent"
- Section 11.5.2.4, "Reverse Proxy Configuration for HTTP PUSH from Mobile Server Not Supported"

11.5.2.1 Proxy Configuration for Web-to-Go Clients

For a Web-to-Go Mobile client, add the proxy server settings as follows in the client `webtogo.ora` file:

```
[WEBTOGO]
PROXY_SERVER=hostname_proxy_server
PROXY_PORT=port_proxy_server
```

11.5.2.2 Proxy Configuration for All Other Clients

For all Mobile clients other than the Web-to-Go Mobile clients, perform the following when you synchronize using the `msync.exe` tool:

1. Check the **Use Proxy** checkbox.
2. Enter the hostname and port number of the proxy server.

11.5.2.3 Proxy Configuration for the Device Management Agent

The Mobile device is behind the firewall and can access the outside world (Mobile Server) only by using a proxy. At the time of client installation, the setup program prompts the user for the proxy information.

If you configured the proxy after the client installation, then you can configure dmagent to use the proxy server by adding `HTTP_PROXY` parameter under the `NETWORK` section including both the IP/hostname and port number to the `polite.ini` file, as follows:

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

11.5.2.4 Reverse Proxy Configuration for HTTP PUSH from Mobile Server Not Supported

In this scenario, the Mobile Server could only initiate communication with a Mobile device behind a firewall through a reverse proxy. However, a reverse proxy would have to be configured for each Mobile device behind the firewall. This is too intensive, so we do not support Mobile Server initiated communication, which includes the HTTP PUSH Device Management communication.

11.6 Branch Office Pre-Installation Considerations

When you install the Branch Office Manager on the Windows machine, it creates the `OracleDatabaseLite` user account with the minimum set of privileges required to execute the Oracle Database Lite software. This prevents Oracle Database Lite Branch Office executing under the `SYSTEM` account, which has broad privileges within the system and can make the system vulnerable.

Both the 'Oracle Lite Multiuser Service' is created as well as the normal Web-to-Go service executes under the privileges of the `OracleDatabaseLite` user. The Oracle Lite Multiuser Server enables remote clients to connect to the Oracle Lite database.

Normally, when installed, the password for the `OracleDatabaseLite` user is randomly generated during the setup. You can either pre-configure this password before the Branch Office installation or modify it after the configuration. See Section 3.5.3, "Defining Password for OracleDatabaseLite User for Branch Office on Windows Machine" in the *Oracle Database Lite Getting Started Guide*.

11.7 Security Warning for Demo Applications

If you have the demo applications installed in a production environment, they can be used to access areas of Oracle Database Lite that you may want to be secure. The demo applications are provided for you to use when learning how to develop your own application. Thus, when you are finished developing your product, remove the demo applications from the repository. For directions, see Chapter 2 "Installation of Oracle Database Lite" in the *Oracle Database Lite Getting Started Guide*.

12

Configure for National Language Support (NLS)

In order to support a language that contains multi-byte characters, perform the following:

- Section 12.1, "Configuring OC4J to Handle Multibyte Characters in Web Applications"

12.1 Configuring OC4J to Handle Multibyte Characters in Web Applications

If you have an application that uses multibyte characters, you need to configure the `default-charset` element to the machine locale in the OC4J `global-web-application.xml` file to allow multibyte characters. For example, a Japanese machine should have its locale set to the `Shift_JIS` locale in the OC4J `global-web-application.xml` file to allow Japanese multibyte characters, as follows:

```
<orion-web-app
   deployment-version="1.0.2.2"
   jsp-cache-directory="./persistence"
   temporary-directory="./temp"
   servlet-webdir="/servlet/"
   default-charset="Shift_JIS">
</orion-web-app>
```

The `global-web-application.xml` file can be found in the *ORACLE_HOME*`/mobile_oc4j/j2ee/home/config` directory. For more information on the elements in the `global-web-application.xml` file, see the *Oracle Application Server Containers for J2EE Servlet Guide*.

13

Reports

This chapter details the types of reports that you can view about the Mobile Server environment:

- Section 13.1, "Viewing System Status Reports for the Server"
- Section 13.2, "Viewing Active User Sessions"

13.1 Viewing System Status Reports for the Server

The Mobile Manager enables users to view system status reports for the Mobile Server. To view system status reports, click the **Administration** link and click the **Summary** link. As Figure 13–1 displays, the Summary page lists Database, JRE, and Operating System details.

Figure 13–1 Summary Page

Summary

Database

Name	Value
Database Version	9.2.0.2.1
Database Name	orcl92
JDBC Driver	Oracle JDBC driver
JDBC Version	9.0.1.5.0
Schema Name	MOBILEADMIN

Java

Name	Value
Java Runtime Environment version	1.4.2_04
Java Runtime Environment vendor	Sun Microsystems Inc.
Java installation directory	C:\Program Files\Java\j2re1.4.2_04

13.2 Viewing Active User Sessions

The Mobile Manager enables administrators to display a list of all users that are connected to the Mobile Server at any given time. To view a report on active user sessions, navigate to the Administration page and click **Sessions**. As Figure 13–2

displays, the Sessions page lists user names, date and time of creating the user's session, and the date and time of the last session.

Figure 13–2 Sessions Page

14

Adding Popular URLs as Bookmarks to Mobile Server Main Page

When you first bring up the Mobile Workspace, before you choose to go to the Mobile Manager, there is a Bookmark tab at the top of the page that lists popular URLs that an administrator has set up. These are URLs that are used often enough that you want to have them easily available.

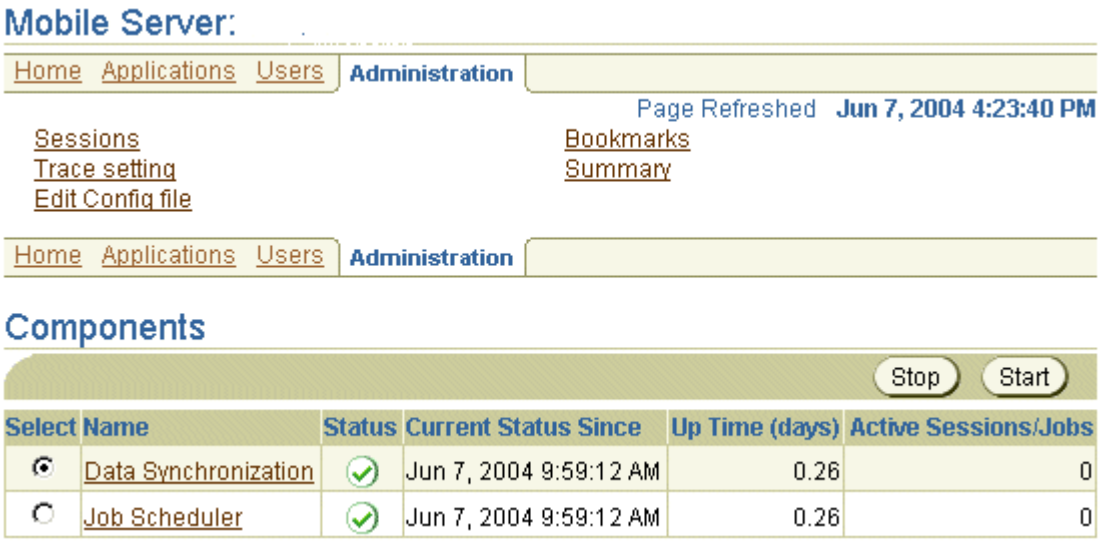
As an administrator, you can set up these bookmarks through the Administration page as detailed in the following sections:

- Section 14.1, "Setting Up Popular URLs as Bookmarks"
- Section 14.2, "Deleting Bookmarks"

14.1 Setting Up Popular URLs as Bookmarks

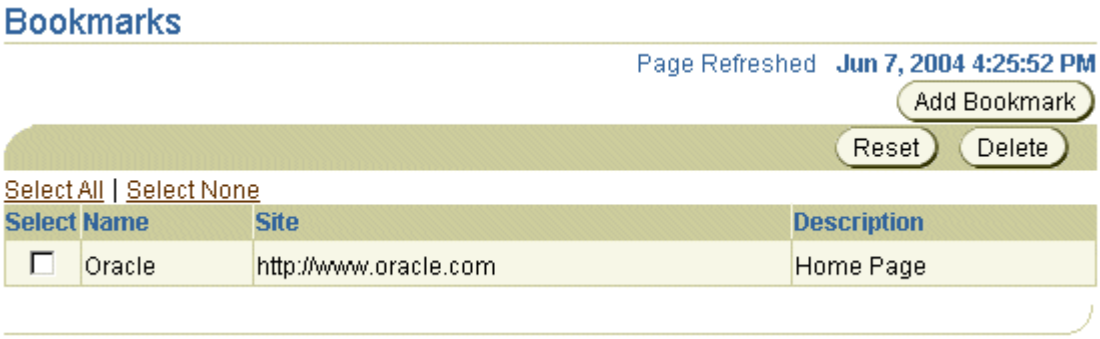
To add bookmarks to popular URLs, click **Administration**, as seen in Figure 14–1.

Figure 14–1 Administration Page



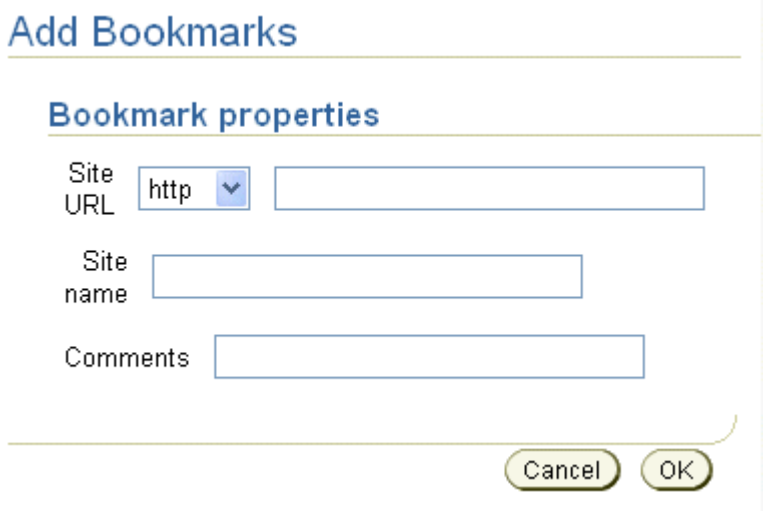
Click **Bookmarks**. As Figure 14–2 displays, the Bookmarks page appears.

Figure 14–2 Bookmarks Page



Click **Add Bookmark**. As Figure 14–3 displays, the Add Bookmarks page appears.

Figure 14–3 Add Bookmarks Page



Enter data under the Bookmark Properties section as described in Table 14–1 and click **Save**. You are returned to the Mobile Server Bookmarks page which lists your bookmark.

Table 14–1 Bookmark Properties Description

Field	Description
Site URL	Web site URL of your Mobile application. Choose the appropriate protocol from the list displayed. For example, to indicate a Web site address, choose http. To indicate a secure Web site address, choose https. To indicate a file transfer site address, choose ftp.
Site Name	Web site name of your Mobile application. For example, www.oracle.com.
Comments	Brief description of the Web site

14.2 Deleting Bookmarks

To delete bookmarks, navigate to the Bookmarks page and select the Bookmark that you want to delete. Click **Delete**.

To reset the bookmarks page, click **Reset**.

A

Configuration Parameters for the WEBTOGO.ORA File

This document describes configuration parameters for the in the `webtogo.ora` file. The Mobile Server uses the `webtogo.ora` file to initialize the Mobile Server; thus, if you modify these parameters, you must restart the Mobile Server to receive the change.

The following sections define system-wide parameters for the Mobile Server:

- Section A.1, "[APPLICATIONS]"
- Section A.2, "[WEBTOGO]"
- Section A.3, "[FILESYSTEM]"
- Section A.4, "[DEBUG]"
- Section A.5, "[PUBLIC]"
- Section A.6, "[SERVLET_PARAMETERS]"
- Section A.7, "[CONSOLIDATOR]"

A.1 [APPLICATIONS]

Table A–1 lists the `APPLICATIONS` parameters and their usage definitions.

Table A–1 APPLICATIONS Parameters

Parameter	Definition
`PACK_HELP`	Location for the Packaging Wizard help file. This is only used with MDK. and only for the purpose of Debug / Support.
`XMLFILE`	Name of the XML file used by the Packaging Wizard to store the application information. This is only used for Debug / Support.

A.2 [WEBTOGO]

The following WEBTOGO parameters control the behavior of both the Mobile client for OC4J or Web-to-Go and the Mobile Server.

Table A–2 lists WEBTOGO parameters and their usage definitions.

Table A–2 WEBTOGO Parameters

Parameter	Definition
ADMIN_PASSWORD	Encrypted user password. Users must not try to edit the encrypted password. This parameter can be set by navigating to the following URL. <server>/webtogo/startup
ADMIN_PORT=8080	Admin port for starting the Mobile Server.
ADMIN_JDBC_ URL=jdbc:oracle:oci8@webtogo.world	Mobile Server Repository JDBC URL.
ADMIN_USER	Encrypted user name. Users must not try to edit the encrypted user name. This parameter can be set by navigating to the following URL. <server>/webtogo/startup
APPLET_USE_THIN_JDBC=YES	Requests that JDBC use the thin driver or the Web-to-Go data communication link for all database calls. Web-to-Go uses the internal Web-to-Go JDBC driver, if it is not using the JDBC thin driver. If this parameter is set to YES, then the parameter THIN_JDBC_URL should also be set.
APPLET_SUPPORT_ENABLE=YES	If you want to run an applet that uses a JDBC connection on the Mobile client for OC4J or Web-to-Go, you must set this parameter to YES and restart the client. If the applet does not use a JDBC connection, you need not set this parameter. Setting this parameter to YES for an applet that does not use a JDBC connection, does not impair your settings.
BIND_IP	Applicable for Web-to-Go Client only. Specify the IP address on which the Web-to-Go listener is bound. By default, the listener is started on the local host. Use this parameter if you have a machine where there is more than one IP address. Define the IP address you wish to use as the Web-to-Go listener and then start the Web-to-Go client listener on that IP address.
CUSTOM_WORKSPACE=no	Indicates whether or not a custom workspace should be used.
CUSTOM_DIRECTORY=/myworkspace	Location of the custom workspace files in the repository.
CUSTOM_FIRSTSERVLET=HelloWorld;/hello	Use this parameter to add the first servlet to the custom workspace. Within the first servlet, you can add more servlets to the custom workspace, using the addServlet() call. Format: class;virtual path
DEFAULT_PAGE=myfirstpage.html	The first page of the custom workspace. This page appears when the user accesses the following URL. http://<server>/webtogo
DEFAULT_CLIENT_1CLICK	The default value for the Mobile Client's "use default setting for sync" Sample Value: YES
DEFAULT_CLIENT_UPGRADE	The default value for the Mobile Client's "ask before upgrade" setting. Sample Value: YES
DEFAULT_CLIENT_SYNCONLY	The default value for the Mobile Client "offline only/online/offline" setting. Sample Value: YES

Table A–2 (Cont.) WEBTOGO Parameters

Parameter	Definition
DISABLE_REMOTE_ACCESS	If set to YES, blocks a remote machine getting access to the Mobile Client for Web or Mobile Client for BC4J. You can set this parameter in the client side webtogo.ora file. Once this parameter is set and Mobile Client is restarted, only the request coming from the local machine is served by the Mobile Client listener. Any other request is blocked and not served. Default value is NO, which is necessary for Branch Office.
DM_AUTO_SYNC_CACHE	Set to YES to turn on this feature to force the DeviceManager to synchronize the cached data on user-related events, such as create, edit or delete user.
ENABLE_USER_ONLINE_ACCESS	Set to TRUE to enable all valid users to log on to the Mobile Server in the Direct Access mode. By default, only users with Administrator privilege level can log in to the Mobile Server in the Direct Mode. Direct mode is when you point the browser directly to the Mobile Server.
FONT_NAME=Arial	The Web-to-Go Workspace font.
IAS_MODE	This parameter must be set to the value YES only if the Mobile Server is running as a component of Oracle9*i*AS. Example: IAS_MODE=YES If the Mobile Server is running in Standalone mode or as a component of Oracle9*i*AS 1.0.2.2.0, this parameter must be set to the value NO. The default value is NO.
INSTALLATION_TYPE	Mobile Server Installation Type, such as STANDALONE, IAS10.1.2.0.0, IAS10.1.3.1.1.0, and so on. Do NOT modify, for internal use only.
IP_CONFIG	Set this parameter in the webtogo.ora file on the Mobile Server to designate what type of IP address the client uses. This parameter specifies if your client device is using static IP address or a dynamic (DHCP) method of retrieving an IP address. If you are using DHCP, then you need to set this parameter to DYNAMIC; the default is STATIC. If you are using DHCP, then the underlying code needs to know to not use the IP address that was used for the previous connection/synchronization. If you are using DHCP and have set this parameter to STATIC, your synchronization may never occur, since it is probably trying to synchronize to an IP address that is no longer valid for this device. Internally, the IP_CONFIG defines if IP address caching occurs in the JVM. Thus, if the IP_CONFIG parameter is set to DYNAMIC, the JVM security property networkaddress.cache.ttl is set to "0", which determines that the JVM always requests a naming service lookup to retrieve the IP address for the client. Disabling Java IP caching may effect performance with the additional DNS lookups. In addition, there is a risk of DNS spoofing. See the Sun Microsystems Java documentation for more information.
LOADBALANCE_URL	If the loadbalancer is used for balancing two or more Mobile Servers, then you may include the LoadBalancer URL with port number. Format is <hostname>:<port>.

Table A–2 (Cont.) WEBTOGO Parameters

Parameter	Definition
MAX_THREAD_POOL	Limits the number of threads available in the connection pool. If threading problems occur, set this parameter to 0 or 1.
MODE=SERVER	The mode the Mobile Server is running in. Valid modes are SERVER, CLIENT, and BRANCH. The value BRANCH indicates that the Mobile Server is running in BRANCH mode for client operations.
OC4J_CONFIG_DIR	Location of the OC4J configuration files that are used by the Mobile Server during runtime.
ORACLE_HOME	Location of the Oracle Home where the Mobile Server is installed.
PORT=80	The port number on which the Mobile Server is running. Not valid in Oracle9*i* Application Server (Oracle9*i*AS) installation.
PROXY_SERVER=proxy.com	The proxy host name and number. The Mobile client for OC4J or Web-to-Go setup modifies this entry.
PROXY_PORT=80	The proxy port number. The Mobile client for OC4J or Web-to-Go setup modifies this entry.
PUBLIC_NAME=/public	The public URLs name. The default value is /public.
REGISTRY_TAB	Turn this parameter on (TRUE) if your application is using the registry setting and you want to manage the registry settings in the Mobile Manager. This parameter enables the registry tab in the Mobile Manager and is only for backward capability.
RESTRICTED_ADMIN_HOSTS=<list of comma separated IP addresses>	This parameter provides security for accounts with Administrator access. With this parameter, the Mobile Server can be configured to allow login requests to a specified set of IP addresses for accounts with Administrator access. With this parameter, you can also restrict access to the Mobile Server Startup feature. Only valid login requests from a browser that runs on machines whose IP address is listed as a value of this parameter will be granted access. For example, RESTRICTED_ADMIN_ HOSTS=144.125.127.150,144.125.127.101 **Note**: Users who have Administrator access should not connect through a proxy server.
REVERSE_PROXY	Set this parameter if a Reverse proxy is used with the Mobile Server. See Section 11.5.1, "Using Reverse Proxy to Communicate from Internet to Intranet" for details.
SERVER_URL=http://<mobile_server_ name>:<port_number>/webtogo	This parameter points to the Mobile Server. It communicates with the Mobile Server over HTTP or HTTPS. Usually, you need not modify this parameter. If you want to run the Mobile client for OC4J or Web-to-Go and download the Mobile client for OC4J or Web-to-Go from the following URL, https://<mobile_server_name>/setup, the Mobile client for OC4J or Web-to-Go is automatically configured for SSL, and no manual configuration is required. The Mobile Client communicates with the Mobile Server over SSL. However, if you do not download the Mobile client for OC4J or Web-to-Go from the Mobile Server that is running in SSL mode and you want to run your Mobile Server in SSL mode, you must modify the SERVER_URL parameter in the configuration file webtogo.ora, on the client side as displayed in the left column.

Table A–2 (Cont.) WEBTOGO Parameters

Parameter	Definition
SSO_ENABLED	Turn this parameter on (TRUE) if you want to enable Oracle Single Sign on (SSO) authentication on the Mobile Server. If this parameter is turned on then the users trying to connect to the Mobile Server in the online mode will receive the login page from the SSO server.
SYNC_CANCEL	This parameter can be set on the client side to determine if the "Cancel" link should appear on the synchronization page. If this parameter is set to YES, the "Cancel" link appears on the synchronization page. By clicking the Cancel link, you can stop the data synchronization. The link will not appear after the data synchronization is complete.
SQL_RETRIES=5	Number of attempts to modify a JDBC connection before timing out.
SSL=YES	If this parameter is set to YES, then the Mobile Server runs in SSL mode. To use this feature, the Mobile Server should be running as a module inside Oracle9*i* Application Server (Oracle9*i*AS).
THIN_JDBC_ URL:jdbcoracle:thin:@foo-pc:1521:orcl	The Mobile Server Repository's thin JDBC URL. This URL is used by the JDBC thin driver to connect to the Mobile Server Repository database.
USE_SYSTEM_CLASSPATH=YES	If set to yes, searches for Java classes in the computer's classpath before searching the Mobile Server Repository.
WTG_PROXY	HTTP proxy used to connect to the Mobile Server for application deployment. Sample Value: www-proxy.dlsun1.com
WTG_PROXY_PORT	HTTP proxy port used to connect to the Mobile Server for application deployment. Sample Value: 80

A.3 [FILESYSTEM]

The following FILESYSTEM parameters control the behavior of the Mobile Server Repository.

Table A–3 lists [FILESYSTEM] parameters and their definitions.

Table A–3 FILESYSTEM Parameters

Parameter	Definition
TYPE	Type of File system. OL - Oracle Lite based file system. OS - Operating system's file system. MIXED - Mixed file system.
PRIMARY=OL	Primary file system in MIXED mode.
SECONDARY=OS	Secondary file system in MIXED mode.

Table A–3 (Cont.) FILESYSTEM Parameters

Parameter	Definition
ROOT_DIR=*ORACLE_HOME*/MOBILE/SERVER/REPOSITORY	Root Directory. Valid only for OS file system. This directory path format applies to the environment where the Mobile Server runs on Solaris. Replace *ORACLE_HOME* with your actual Oracle Home.
ROOT_DIR=*ORACLE_HOME*\MOBILE\SERVER\REPOSITORY	Root directory. Valid only for OS file system. This directory path format applies to the environment where the Mobile Server runs on Windows NT. Replace *ORACLE_HOME* with your actual home.

A.4 [DEBUG]

The following DEBUG parameters control the debugging messages in the Mobile Server.

Table A–4 lists DEBUG parameters and their definitions.

Table A–4 DEBUG Parameters

Parameter Name	Definition
TRACE_ENABLE	Used to turn the trace feature on or off. When the Trace feature is off, trace output is not generated. This value is only overridden when the Mobile Server is running in Standalone mode and with the -d0 command line option on. For example: TRACE_ENABLE=NO Is overridden by -d0 and the trace output is generated to the Console instead of being generated to a file. Sample Value: YES
TRACE_DESTINATION	Trace destinations are Console and File. The Administrator can set this parameter to any of these destinations. The Console option generates trace output to the console screen. **Note**: This trace destination is available only when the Mobile Server is running in Standalone mode. If you set this parameter to the option -d0, the trace output appears on your Console window without appearing in a file, because using the -d0 option with this parameter overrides the trace settings for other trace parameters, such as destination and level, in the webtogo.ora file. The -d0 setting enforces the trace output to appear on your console screen instead of appearing in a file. The File option generates trace output to a file. For more information, see TRACE_FILE_NAME, TRACE_FILE_SIZE, and TRACE_FILE_POOL_SIZE. Sample Value: TRACE_DESTINATION=FILE

Table A–4 (Cont.) DEBUG Parameters

Parameter Name	Definition
TRACE_FILE_NAME=trace.log	Used as base name to arrange trace files in sequential order starting from 1 to FILE_ TRACE_POOL_SIZE. For example: If you set the following parameters. TRACE_FILE_NAME=mytrace.log TRACE_FILE_POOL_COUNT=5 then, the Trace files will be named mytrace1.log, mytrace2.log, mytrace3.log, mytrace4.log, mytrace5.log, based on how you set the TRACE_FILE_PER_USER parameter. Sample Value: trace.log
TRACE_LEVEL=1	There are three levels of trace messages: 1(binary 00000001), Basic Trace: General system information, most of the Web-To-Go trace output belongs to this level. 2 (binary 00000010), Function Trace: Traces the function sequence being called, mostly used by Data Synchronization. 4 (binary 00000100), SQL Trace: Traces SQL queries being executed, mostly used by Data Synchronization. In addition, all errors and exceptions are sent to level -1, which have the binary 11111111. All Java System.out output are sent to level 9. Both these two levels are always generated as output, if the user is not filtered out. For more information, see TRACE_USER. The parameter value for TRACE_LEVEL is used to do a Bitwise AND operation against all 3 trace levels. If the result is greater than 0, then trace output of that level will be generated as trace output. The parameter value for TRACE_LEVEL used to do a Bitwise AND operation against all 3 trace levels. If the result is greater than 0, then trace output of that level will be generated as trace output. EXAMPLE: If you set the following parameters, TRACE_LEVEL=3, then the Basic and SQL level trace output is generated, but not Function level trace as the & character is a Bitwise AND operator. 3 & 1 (Basic) = 1 > 0 3 & 2 (SQL) = 2 > 0 3 & 4 (Function) = 0 = 0

Table A–4 (Cont.) DEBUG Parameters

Parameter Name	Definition
TRACE_USERS	List of valid user names. The user trace and system trace information which is listed is generated as trace output. If the value is an empty string "", then every user is traced. If the value is or contains TRACE_NO_USER, then no actual user is traced. Only the system trace information is generated as trace output. **Note**: As the administrator, you must not use the TRACE_NO_USER value as the user name. Example: If you set this parameter as follows, TRACE_USERS=jane,jack, then only jane and jack's trace information is generated and displayed as trace output.
TRACE_FILE_PER_USER=YES	Used to specify an individual trace file pool for every individual user. Applicable only when the **File** option is the Trace destination. If set to YES, then every traceable user has an own trace file pool, and the trace file name includes the user's name. In addition, the system trace output goes to the user's system trace file. If set to NO, all traceable users share the same trace file pool, the actual trace file does not contain any user name. Example: TRACE_FILE_POOL_PER_USER=No
TRACE_FILE_SIZE=10	Used as the maximum file size in MB for trace files. If the threshold value is about to be reached, the trace feature generates output to the next trace file in the pool. For more information, see TRACE_FILE_NAME, TRACE_FILE_POOL_SIZE, and TRACE_FILE_PER_USER.
TRACE_FILE_POOL_SIZE=5	The default value is 5. This parameter specifies the number of files in the trace file pool. If the pool limit is reached, the trace output is overwritten to the first file in the pool. See also TRACE_FILE_NAME, TRACE_FILE_POOL_SIZE, and TRACE_FILE_PER_USER

A.5 [PUBLIC]

The following PUBLIC parameters control public availability of servlets in the Mobile Server. To make a servlet public, you can use the parameters as listed in the following table.

Table A–5 lists PUBLIC parameters and their definitions.

Table A–5 PUBLIC Parameters

Parameter Name	Definition
myservlet=/<virtualpath>	To call this public URL from your application, call it as follows: http://<server>/public/<virtual path> For example, oracle.codeMyservlet=/my servlet oracle.codeMyservlet=/myservlet

A.6 [SERVLET_PARAMETERS]

In the SERVLET parameters section, you can list the set of custom parameters which are available to all servlets inside the Mobile Server.

Table A–6 lists SERVLET_PARAMETERS and their definitions.

Table A–6 SERVLET Parameters

Parameter Name	Definition
MY_VAR=MY_VALUE	Custom parameter which can be accessed by all servlets.

A.7 [CONSOLIDATOR]

The CONSOLIDATOR parameters control the behavior of Data Synchronization. The values that are listed in the following table are default values.

- Section A.7.1, "Data Synchronization Parameters"
- Section A.7.2, "Data Synchronoization Tracing and Logging"

A.7.1 Data Synchronization Parameters

Table A–7 Consolidator Parameters

Parameter Name	Definition
APPLY_TRIES_DELAY	Specifies in seconds the delay between successive attempts to apply a client's In Queue. Related to the MAX_APPLY_TRIES parameter.
CACHED_USERS	Comma-separated list of users, where each user's downloaded data is cached.
CLIENT_RESEND_CHECK	If set to YES, then all client resend sessions requests are rejected. This is the default. If a client uploads a transaction and the server receives and processes it correctly, but the client is not notified, then the client will send the transaction again. To avoid having this transaction be processed again, the resend session request is rejected. However, if you are developing an application and you are executing a test, such as a performance test, you will resend the same client request to the server repeatedly to test how the server responds under heavy load. Only in this case would you want to set this parameter to NO.

Table A–7 (Cont.) Consolidator Parameters

Parameter Name	Definition
COMPOSE_TIMEOUT=300	Specifies in seconds the MGP timeout for the compose phase for each user to complete. If the compose phase for this user does not complete, MGP retries the compose phase for this user in the next cycle. If the compose consistantly fails then increase timeout value. Monitor the MGP logging to evaluate how long the compose takes to complete; then add 50% to the value ensure that slightly larger datasets compose completely.
CONN_CHECK_ON_RESERVE	Configure whether to validate a database connection before retrieving (borrowing) it from the connection pool. By default, this value is YES.
CONN_CHECK_ON_RELEASE	Configure whether to validate the database connection before releasing it back to the connection pool. By default, this value is YES. If the examination determines that the connection is not valid, then it is destroyed instead of being returned to the connection pool.
CONNECTION_POOL=YES	Enables pooling of database connections if set to YES.
CONNECTION_TIMEOUT=120	Specifies in minutes the JDBC connection timeout for the synchronization session. If synchronization takes longer than the value specified in this parameter, then the server can automatically disconnect. To avoid the connection from timing out during a valid synchronization, then set this value higher.
DO_APPLY_BFR_COMPOSE	By default, before the MGP processes the Compose phase for a user, it checks to see if user data has been uploaded into the In Queue. If so, then the Compose is not performed to ensure that user data is not overwritten. Instead, the Compose phase is not executed until the MGP runs the Apply/Compose phase again. Setting DO_APPLY_BFR_COMPOSE to true modifies this behavior. If data for a user is in the in queue, MGP will execute a second Apply to commit all user data and then will execute the Compose for that user.
IN_QUEUE_INDEX_ATTRIBUTES	Specify the index attributes for the In Queue table indexes, which exist in the back-end Oracle database. These can include the storage characteristics and the following attributes: TABLESPACE, PCTFREE, PCTUSED, INITRANS, and MAXTRANS. See the *Oracle Database SQL Reference* for more information on these properties.
IN_QUEUE_TABLE_PROPERTIES	Specify the physical and/or table properties for the In Queue tables, which exist in the back-end Oracle database. These can include the storage characteristics and the following properties: TABLESPACE, PCTFREE, PCTUSED, INITRANS, and MAXTRANS. See the *Oracle Database SQL Reference* for more information on these properties.
JDBC_URL	This is the JDBC_URL used by the Sync Service and the MGP for connections to the Mobile Server Repository. If absent, it defaults to the ADMIN_JDBC_URL in the WEBTOGO section of the webtogo.ora file.
JOB_ENGINE_AUTO_START	If set to YES, the Job Scheduler is started up at Mobile Server start time.
JOB_ENGINE_SLEEP_TIME	The amount of time in seconds that the Job Scheduler sleeps in each loop.

Table A–7 (Cont.) Consolidator Parameters

Parameter Name	Definition
LOG_LOCK_DELAY	Specifies in the number seconds the delay between successive attempts to lock the log tables. Related to the MAX_LOG_LOCK_TRIES parameter.
MAGIC_CHECK	Control the magic number checking of publication items. If enabled, and there is a mismatch between the server and the client magic numbers, then the publication item receives a complete refresh. If set to ALL, then the magic check is enabled, which is the default. If NONSHARED, then the magic check is enabled only for publication items that are not shared among users. if set to SHARED, then the magic check is enabled only for shared publication items. A shared publication item has the following characteristics: ■ Read only ■ The publication item query either has no parameters or all users share the same parameter values. Setting to NONSHARED is useful when creating a installation CD for users that share data. See Section 9.8, "Creating a Single Package or Shared CD for Users That Share Data" for more information.
MAP_INDEX_ATTRIBUTES	Specify the index attributes for the map table indexes, which exist in the back-end Oracle database. These can include the storage characteristics and the following attributes: TABLESPACE, PCTFREE, PCTUSED, INITRANS, and MAXTRANS. See the *Oracle Database SQL Reference* for more information on these properties.
MAP_TABLE_PROPERTIES	Specify the physical and/or table properties for the map tables, which exist in the back-end Oracle database. These can include the storage characteristics and the following properties: TABLESPACE, PCTFREE, PCTUSED, INITRANS, and MAXTRANS. See the *Oracle Database SQL Reference* for more information on these properties.
MAX_APPLY_TRIES	Specifies the maximum number of times the MGP retries to apply the client In Queue data before terminating the apply phase.
MAX_BATCH_SIZE	JDBC performance parameter which defines the number of DML records (inserts only) to send from the Mobile Server to the back-end Oracle database at one time. Without batching, each record is sent to the database one at a time. These records are originally from client. This only applies to insert, and not to delete or update records.
MAX_CONNECTIONS=1000	Sets the maximum number of JDBC connections that can be open at one time by the Mobile Server. When this number is reached, no further synchronization sessions are allowed until active connections are released back to the connection pool.

Table A–7 (Cont.) Consolidator Parameters

Parameter Name	Definition
MAX_CONCURRENT	Sets the maximum number of concurrent active synchronization sessions. When this number is reached, subsequent synchronization session requests are placed in a FIFO queue, and are only allowed to continue when active sessions complete and slots become available. This parameter is designed to prevent the Mobile Server from being overloaded by concurrency and can help to improve throughput when the hardware is being stressed. Active sessions are concurrently processed and the sessions where users are waiting to be processed are placed in a FIFO queue. The MAX_CONCURRENT variable is the maximum number of active sessions. If after the MAX_CONCURRENT_TIMEOUT, the number of active sessions is still at the maximum, then the waiting sessions that are older than the MAX_CONCURRENT_TIMEOUT are disconnected. Then, the client receives an error that the synchronization has failed. The Oracle Lite client resends the same request during next synchronization, which consists of a push-only resend for the transaction that failed and another synchronization for any new uploaded data.
MAX_CONCURRENT_TIMEOUT	Sets the maximum number of seconds to wait
MAX_LOG_LOCK_TRIES	Specifies the number of attempts to lock the logs before giving up the compose phase.
MAX_THREADS=3	Specifies the number of threads spawned within the MGP process. This parameter value should be set to an equivalent number of CPUs. We recommend this value is set to 1.5 times the number of CPUs. The default is 3.
MGP_CYCLES_BEFORE_INS_CHK	How often should MGP compose phase look for map table records that are marked for deletion but have been re-added to the subscription. By default, this is set to 1. If the application logic guarantees that such records never exist, then this check may be skipped altogether by setting the value to –1, which improves MGP compose performance.
MGP_HISTORY	If set to YES, enables MGP history recording, which can be viewed in the Data Synchronization section of the Mobile Manager.
MGP_PORT	The IP port used by MGP to report current state and/or coordinate multiple instances of the process. The default port number is 7373.
REPORT_ALL_ERRORS	If set to YES, then the MGP attempts to detect and report all apply transaction errors. If set to NO, which is the default, only the first error is reported.
RESUME_CLIENT_MAXSEND	The RESUME_CLIENT_MAXSEND parameter is the maximum data size, in KB, that the client should send in a single POST request. This is used in cases where there is a proxy with a small limit on the data size in one request. Specifying a reasonable value, such as 256 KB, can also help clients with limited storage space, as they can free the chunks that have already been transmitted and acknowledged. The default is 1024 KB. Set the maximum data size in KiloBytes sent by a client in a single POST request. Some proxies maintain fixed limits on data size in one request.

Table A–7 (Cont.) Consolidator Parameters

Parameter Name	Definition
RESUME_CLIENT_TIMEOUT	The RESUME_CLIENT_TIMEOUT parameter is the number of seconds that the client should use to timeout network operations. The default is 60 seconds. Set the total number of seconds that the client should use to resume network timeout operations.
RESUME_FILE	This is a server-side parameter that defines the filename of the resume buffer. By default, we use memory mapped file; however, with this parameter, users can provide their own storage files. The size of this file is specified by the RESUME_FILE_SIZE parameter.
RESUME_FILE_SIZE	Set the maximum size of the resume file in MegaBytes (MB).
RESUME_MAXACTIVE	The RESUME_MAXACTIVE parameter controls the maximum number of connections taht the Mobile Server handles at a single time. If more clients try to connect, they are queued until existing connections complete. The default is 100 connections.
RESUME_MAXCHUNK	The RESUME_MAXCHUNK parameter causes the server to drop the connection after sending the specified data size, in KB. This forces the client to reconnect and inform the server on how much data it already has. The server can the discard all data before that offset. The fault value is 1024 KB.
RESUME_TIMEOUT	The RESUME_TIMEOUT parameter indicates how long to keep client data while the client is not connected. The default is 0, which means that resume is disabled and after disconnection, the client data is discarded. A short timeout, such as 15 minutes, is suitable to resume any accidentally dropped connections. A longer timeout may be needed if users explicitly pause and resume synchronization to switch networks or use a dialup connection for another purpose.
SKIP_INQ_CHK_BFR_COMPOSE	By default, before the MGP processes the Compose for a user, it checks to see if user data has been uploaded into the In Queue. If so, then the Compose for the user is not performed to ensure that user data is not overwritten. Instead, the Compose phase is not executed until the MGP runs the Apply/Compose phase again. Setting SKIP_INQ_CHK_BFR_COMPOSE to true modifies this behavior. Even if data is in the in queue, MGP executes the Compose for the user. The data that was uploaded to the In Queue must be data that will not be compromised by downloading data from the server to the client.
SLEEP_TIME=20000	Specifies how long (in milliseconds) the MGP sleeps before scheduling the next client's apply phase. By default, it is set to 2 milliseconds.
STMT_CACHE_SIZE	The number of prepared statements to be cached for each connection. These statements are not re-parsed by the database when they are prepared again. To turn off caching, set this variable to -1.
SYNC_HISTORY	If set to YES, enables synchronization history recording, which can be viewed in the Data Synchronization section of the Mobile Manager.

Table A–7 (Cont.) Consolidator Parameters

Parameter Name	Definition
SYNC_SERVICE_AUTO_START	If set to YES, the Sync Server is started automatically at Mobile Server startup. The client cannot synchronize until the Sync Server is started. So, if you want to prevent the client from synchronizing, then set this parameter to NO and then start and stop the Sync Server manually through the Mobile Manager after you have completed the tasks that you want to perform while the client is unable to synchronize. For example, you do not want a client to synchronize if you are re-publishing the client's application.
TEMP = C:\TEMP	Specifies the directory where the binary trace file, which includes the request and response data, is written. This file is created to cache the data in transport, so that if a failure occurs, Oracle Lite can recover. You initialize this binary trace file by selecting the Data Trace Type checkbox.
USE_JVM_COMPRESSION	Specifies whether the JVM implementation of ZIP or the Oracle pure Java version for compression should be used. This is a performance parameter, where depending on a particular environment, each implementation may have different performance results. By default, this is set to YES to use the JVM implementation.

A.7.2 Data Synchronoization Tracing and Logging

Data Synchronization uses a log engine that supports the following parameters for logging:

- GLOBALLogger
- SYNCLogger
- MGPLogger
- MGPAPPLYLogger
- MGPCOMPOSELogger

Each parameter sets up a logger for a component which you can use to specify the trace level, trace type, trace destination, trace file pool size, trace file size, and trace users in the following sample format.

XLogger=TRACE_LEVEL=<*trace_level*>|TRACE_TYPE=<*trace_type[,trace_type...]*>|TRACE_DESTINATION=<*trace_destination*>[|TRACE_FILE_POOL_SIZE=<*trace_file_pool_size*>|TRACE_FILE_SIZE=<*trace_file_size*>|TRACE_USER=<*trace_users*>]

Note:

- Separate each parameter with the '|' symbol. Separate values with a comma ',' .
- If there are any invalid values in the definition, the whole definition is ignored.
- For each logger, the trace level, type, and destination parameters are mandatory.
- The parameters TRACE_FILE_POOL_SIZE and TRACE_FILE_SIZE are only applicable for the GLOBALLogger only.
- If you define the LOCAL_CONSOLE, then you must also define SYNCLogger and GLOBALLogger.

Table A–8 lists the parameters for each logger.

Table A–8 Acceptable Parameter Values

Parameter	Description
TRACE_LEVEL	Trace Level parameter can be set to the following trace message levels: MANDATORY: This option logs mandatory messages only. For example, Program Exceptions. Regardless of component settings, this option logs exceptions in the error log file (err.log) located in the Conslog directory. WARNING: This option logs warning messages and messages at the Mandatory level. For example, Program Exceptions that users can ignore, messages that the program wants to warn the users with, and so on. NORMAL: This option logs normal messages that the user must be informed with and messages at the Mandatory and Warning level. INFO: This option logs information messages and messages at the Mandatory, Warning, and Normal levels. Examples: ■ Timing of synchronization: When the SYNCLogger is set to the TRACE_TYPE=TIMING and TRACE_LEVEL=INFO. ■ MGP Apply: When the MGPAPPLYLogger is set to the TRACE_TYPE=TIMING and TRACE_LEVEL=INFO. MGP Apply must be started with Timing of. COMPOSE must be started with Timing of MGP Compose. MGP must be started with Timing of. ■ COMPOSE: When the MGPAPPLYLogger is set to the TRACE_TYPE=TIMING and TRACE_LEVEL=INFO. ■ Status of MGP: When the MGPLogger is set to the TRACE_TYPE=GENERAL and TRACE_LEVEL=INFO. CONFIG: This option logs configuration messages and messages at the Mandatory, Warning, Normal, and Info levels. For example, JDBC driver version. FINEST: The finest level. This level is used for developers only. ALL: This option logs all messages according to the other settings such as Trace Type and Users.

Table A–8 (Cont.) Acceptable Parameter Values

Parameter	Description
TRACE_TYPE	SQL: This option logs SQL-related messages only. For example, SQL statements. **Note**: This option is not trace level sensitive.
	TIMING: This option logs timing data only. **Note**: This option is trace level sensitive. For MGP Cycle time and Synchronization time, use the Trace Level INFO option. If the MGPLogger is set to TIMING and INFO, it will log the MGP Cycle time. If the SYNCLogger is set to TIMING and INFO, it logs the synchronization time.
	DATA: This option logs data only. **Note**: This option is not trace level sensitive. This option prints all data with any trace level other than the OFF option.
	RESUME: Messages dealing with Reliable Transport have a RESUME trace type. This option only logs messages with Reliable Transport. **Note**: This option is not trace level sensitive. This option prints all the RESUME trace type messages with any trace level other than the OFF option.
	FUNCTION: This option displays the program flow by logging methods such as Entry, Exit or Invoke. For Long methods, this option logs the method's entry or exit; which is a simple invoke log. **Note**: This option is not trace level sensitive. This option prints all the FUNCTION trace type messages with any trace level other than the OFF option.
	GENERAL: This option logs messages that do not belong to any of the above listed trace types. **Note**: This type is trace level sensitive.
	ALL: This option generates logs of all trace types.
TRACE_DESTINATION	The Administrator can set this parameter to any of these destinations: LOCAL_CONSOLE or TEXTFILE. The Console option generates trace output to the Console screen. The TEXTFILE option generates trace output to a file. See also TRACE_FILE_SIZE, and TRACE_FILE_POOL_SIZE.
	Sample Value: TRACE_DESTINATION=TEXTFILE
TRACE_FILE_POOL_SIZE=2	The default value is 2. This parameter specifies the number of files in the trace file pool. If the pool limit is reached, the trace output is overwritten to the first file in the pool. See also TRACE_FILE_POOL_SIZE.
TRACE_FILE_SIZE=1	Used as the maximum file size in MB for trace files. If the value is about to be reached, the trace feature generates output to the next trace file in the pool. For more information, see TRACE_FILE_POOL_SIZE.
TRACE_USERS	List of valid user names. The listed user trace information and system trace information is generated as output. If the value is an empty string "", then every user is traced.

The new log engine does not support the parameters that have been used in the old log engine. They are:

- TRACE_ENABLE
- TRACE_REMOTE_PORT
- TRACE_REMOTE_MACHINE
- TRACE_FILE_PER_USER
- TRACE_FILE_NAME

- TRACE_REMOTE_HOST

B

Scripting Language for the Mobile Server

The following sections describe the scripting language for the Mobile Server. You can use scripting to perform batch processing tasks that are performed frequently by the administrator. You can write scripts for the Mobile Server in an INI text file and use the `WSH` tool to run your INI script.

B.1 Description of the Syntax

The following sections describe the parameters and syntax available in the scripting language:

Creating a User

Using the following syntax, you can create users to be included in a group.

```
[USER]
NAME=<User Name>
PASSWORD=<User Password>
```

```
ENCRYPTED=<True or False; True if the password is encrypted, False if not>
FULLNAME=<User Full Name>
PRIVILEGE=<User privilege level as P, C, S, or null>
```

There are four options for setting the PRIVILEGE value for users. They are:

- **P** - Publishing an application
- **C** - Connecting to Web-to-Go
- **S** - Manage Web-to-Go
- **Null** - No privileges

Creating a Group

Using the [GROUP] script, you can create a new group (if this group does not already exist) and add listed users to the group. If you use this entry and specify the name of a group that exists, all the users in the existing group will be removed and users who are listed will be added to this group.

The following syntax enables you to create a group.

```
[GROUP]
NAME=<Group Name>
USER=<User name you want to add to this group>
USER=<User name you want to add to this group>
USER=<User name you want to add to this group>
```

Adding Users to a Group

Using the [ADDUSERTOGROUP] script, you can create a new group (if this group does not already exist) and add listed users to this group. You can also use this entry to add users to an existing group.

```
[ADDUSERTOGROUP]
NAME=<Group Name>
USER=<User name you want to add to this group>
USER=<User name you want to add to this group>
```

Removing Users from a Group

Using the [REMOVEUSERFROMGROUP] script, you can remove listed users from a specified group.

```
NAME=<Group Name>
USER=<User name you want to remove from this group>
USER=<User name you want to remove from this group>
```

Creating Access Privileges

Using the [ACL] script, you can create a new ACL (if this ACL does not already exist). After creating the ACL, all the existing users will be removed and all the listed users will be added to this ACL.

Using the [GRANTACCESS] script, you can add users to the existing ACL.

The following syntax enables you to create access privileges for users and groups.

```
[ACL]
APPLICATION=<Name of the application you want to creat ACL for>
ROLE=<Role of the user; set the value as DEFAULT ROLE or ADMINISTRATIVE ROLE>
USER=<User's name>
ACCESS=<Set access status as ENABLED>
ROLE=<Role of the user>
```

```
USER=<User name>
ACCESS=<Set access status as ENABLED>
ROLE=<Role of the group>
GROUP=<Groups name>
ACCESS=<Set access status as ENABLED>
```

Granting Access

Using the `[GRANTACCESS]` script, you can create a new ACL (if this ACL does not already exist) and add listed users to this ACL.

```
[GRANTACCESS]
APPLICATION=<Name of the application you want to add ACL for>
ROLE=<Role of the user>
USER=<User name>
ACCESS=<Access Status ENABLED/DISABLED>
ROLE=<Role of the group>
GROUP=<Group name>
```

Revoking Access

Using the `[REVOKEACCESS]` script, you can remove users that are listed in the specified ACL.

```
[REVOKEACCESS]
APPLICATION=<Name of the application you want to revoke ACL for>
ROLE=<Role of the user>
USER=<User name>
ACCESS=<Access Status>
ROLE=<Role of the group>
GROUP=<Groups name>
```

Creating Registries

Using the `[REGISTRY]` script, you can create registries.

```
[REGISTRY]
APPLICATION=<Name of the application>
NAME=<Registry Variable Name>
VALUE=<Value for this variable>
```

Creating Snapshot Variables

Using the `[SNAPSHOTVAR]` script, you can create snapshot variables.

```
[SNAPSHOTVAR]
NAME=<Name of the publication item>
PLATFORM=<Platform for which this publication item is>
VIRTUALPATH=<Virtual path of the application this publication item
belongs to>
USER=<Name of the user who subscribes to this application>
VAR=<Name of the Data Subsetting parameter, value of this parameter>
USER=<Name of the user who subscribes to this application>
VAR=<Name of the Data Subsetting parameter, value of this parameter>
GROUP=<Name of the group which subscribes to this application>
VAR=<Name of the Data Subsetting parameter, value of this parameter>
```

Deleting a User

Using the `[DROPUSER]` script, you can delete a user.

```
[DROPUSER]
NAME=<User Name>
```

Deleting a Group

Using the [DROPGROUP] script, you can delete a group.

```
[DROPGROUP]
NAME=<Group Name>
```

Deleting Access Privileges

Using the [DROPACL] script, you can delete access privileges provided to users.

```
[DROPACL]
APPLICATION=<Name of the application you want to delete ACL for>
ROLE=<Role of the user; set the value as DEFAULT ROLE or ADMINISTRATIVE ROLE>
USER=<User name>
ACCESS=<Set access status as DISABLED>
ROLE=<Role of the group; set the value as DEFAULT ROLE or ADMINISTRATIVE ROLE>
GROUP=<Groups name>
ACCESS=<Set access status as DISABLED>
```

Deleting a Registry

Using the [DROPREGISTRY] script, you can delete a registry.

```
[DROPREGISTRY]
APPLICATION=<Name of the application>
NAME=<Registry Variable Name>
VALUE=<Value for this variable>
```

Deleting Snapshot Variables

Using the following [DROPSNAPSHOTVAR] script, you can delete snapshot variables.

```
[DROPSNAPSHOTVAR]
NAME=<Name of the publication item>
PLATFORM=<Platform for which this publication item is>
VIRTUALPATH=<Virtual path of the application this publication item belongs to>
USER=<Name of the user who subscribes to this application>
VAR=<Name of the Data Subsetting parameter, value of this parameter>
USER=<Name of the user who subscribes to this application>
VAR=<Name of the Data Subsetting parameter, value of this parameter>
GROUP=<Name of the group which subscribes to this application>
VAR=<Name of the Data Subsetting parameter, value of this parameter>
```

B.2 Running a Script INI File

To run a script INI file using the WSH tool, use the following command:

```
WSH -c <filename.ini> mobileadmin/manager@webtogo.world
```

B.3 Examples

The following sections enable you to accomplish the following tasks and describes examples from a script file in INI format:

- Section B.3.1, "Creating, Adding, and Granting Access"
- Section B.3.2, "Deleting, Removing, and Revoking Access"

B.3.1 Creating, Adding, and Granting Access

The following examples illustrate how to create users, groups, registries, access privileges, snapshotvar template variables, add users to a group, and add users to an ACL.

```
[DATABASE]
TYPE=ORACLE
#Creation or modification of users, groups, access privileges, registry,
and snapshot variable entries using the following entries in the INI file:
#[USER], [GROUP], [ACL], [REGISTRY],[SNAPSHOTVAR].
# Create user JOHN
#
[USER]
NAME=JOHN
PASSWORD=john
ENCRYPTED=false
FULLNAME=Sample1 User John
PRIVILEGE=C
#
# Create group 'Sample Users' contaning JANE, JOHN, JACK
#
[GROUP]
NAME=Sample Users
USER=JANE
USER=JOHN
USER=JACK
#
# Set the ACL on the Sample3 application.
# The following gives John, Jane, and Jack, plus all the users in the group
# Sample Users access to the application
#
[ACL]
APPLICATION=/sample3
ROLE=Default Role
USER=JOHN
ACCESS=ENABLED
ROLE=Default Role
USER=JANE
ACCESS=ENABLED
ROLE=Default Role
USER=JACK
ACCESS=ENABLED
ROLE=Default Role
GROUP=Sample Users
ACCESS=ENABLED
#
# Add registry entry for user JOHN and a default value for the Sample3
application to the Web-to-go Repository
#
[REGISTRY]
APPLICATION=/sample3
USER=JOHN
NAME=USERCODE
VALUE=1111
#
# Add template variables.
# You can specify user/group specific values for these variables
#
[SNAPSHOTVAR]
NAME=RECORDINGS
```

```
PLATFORM=WIN32
VIRTUALPATH=/sample3
USER=JOHN
VAR=CODE, 1111
USER=JACK
VAR=CODE, 1111
USER=JANE
VAR=CODE, 2222
GROUP=Sample Users
VAR=CODE, 2222
#
#Add users to a group.
#
[ADDUSERTOGROUP]
NAME=Sample Users
USER=USER1
USER=USER2
#
#Grant Access to users.
#
[GRANTACCESS]
APPLICATION=/sample3
ROLE=Default Role
USER=USER1
ACCESS=ENABLED
ROLE=Default Role
USER=USER2
ACCESS=ENABLED
ROLE=Default Role
GROUP=Sample Users
```

B.3.2 Deleting, Removing, and Revoking Access

The following examples illustrate how to delete a user, group, registry and snapshotvar, remove users from a group, and revoke access.

```
#Deletion of users, groups, access privileges, registry and snapshot
#variable entries using the following entries in the INI file:
#[DROPUSER], [DROPGROUP], [DROPACL], [DROPREGISTRY],[DROPSNAPSHOTVAR].
#
# Dropuser JOHN
#
[DROPUSER]
NAME=JOHN
#
# Drop group 'Sample Users'
#
[DROPGROUP]
NAME=Sample Users
#
# Drop the ACL on the sample3 application.
#
[DROPACL]
APPLICATION=/sample3
ROLE=Default Role
USER=JOHN
ACCESS=DISABLED
ROLE=Default Role
GROUP=Sample Users
ACCESS=DISABLED
```

```
#
# Drop registry entriy for user JOHN from Sample3 application.
#
[DROPREGISTRY]
APPLICATION=/sample3
USER=JOHN
NAME=USERCODE
#
# Drop template variables for user JOHN and group 'Sample Users'
#
[DROPSNAPSHOTVAR]
NAME=RECORDINGS
PLATFORM=WIN32
USER=JOHN
VAR=CODE, 1111
GROUP=Sample Users
VAR=CODE, 2222
#
#Remove users from a group.
#
[REMOVEUSERFROMGROUP]
NAME=Sample Users
USER=USER1
USER=USER2
#
#Revoke access.
#
[REVOKEACCESS]
APPLICATION=/sample3
ROLE=Default Role
USER=USER1
ACCESS=DISABLED
ROLE=Default Role
USER=USER2
ACCESS=DISABLED
ROLE=Default Role
GROUP=Sample Users
```

C

Data Synchronization Requirements in INIT.ORA

The following sections describe the Data Synchronization requirements for Oracle and Oracle parameter settings in the `init.ora` file:

- Section C.1, "Relationships Between Relevant Parameters"
- Section C.2, "Values for Processes and DML Locks"

C.1 Relationships Between Relevant Parameters

You should set the following parameters in the file `init.ora` as given below:

Table C–1 lists parameters that must be set in the file `init.ora`:

Table C–1 init.ora Parameter Settings

Parameter Name	Definition
`PROCESSES`	Default value: 59 to 200.
`SESSIONS`	Default value: Derived: `1.1 * PROCESSES + 5`
`TRANSACTIONS`	Default value: Derived: `(1.1 * SESSIONS)`
`DML_LOCKS`	Default Value: Derived: `(4 * TRANSACTIONS)`

C.2 Values for Processes and DML Locks

Check values for processes and `DML_LOCKS`. Massive concurrent synchronization processes use the maximum amount of resources. For each one of the concurrent clients, Data Synchronization requires one database connection (one session, one transaction). Therefore, the parameter value of `PROCESSES` must be set to be no less than the maximum number of concurrent clients.

During the sync, the Data Synchronization will make changes to the publication map tables. One DML lock is needed for each client and changed publication:

`DML_LOCKS` = (Number of changed publications) * (Maximum number of concurrent clients)

During the first and second sync, all publication map tables are changed for each client. So, the required DML locks are:

DML_LOCKS = (Number of publications) * (Maximum number of concurrent clients)

If you have a large number of publications, the default DML_LOCKS may not be sufficient. You should set it explicitly in the file init.ora. For example, CRM has approximately 50 publications. For 30 concurrent first syncs, Data Synchronization needs 1500 DML locks. The default value for DML_LOCKS with PROCESSES set to 200 is 1000.

D

Catalog Views for the Mobile Server and the Mobile Client

The following sections describe the catalog views for the Mobile Server and the Mobile client:

- Section D.1, "Mobile Server System Catalog Views"
- Section D.2, "Client Oracle Lite Database System Catalogs"

D.1 Mobile Server System Catalog Views

The following sections are a reference for the system catalog views for the Mobile Admin schema. These sections list and describe the complete set of catalog views for the Mobile Server.

The Mobile Admin schema is installed as part of the Mobile Server during installation. However, the Mobile Admin schema is not part of the Mobile Development Kit.

The system catalog views are read-only and should not be modified.

- Section D.1.1, "CV$ALL_CLIENTS"
- Section D.1.2, "CV$ALL_ERROR"
- Section D.1.3, "CV$ALL_PUBLICATIONS"
- Section D.1.4, "CV$ALL_SUBSCRIPTIONS"
- Section D.1.5, "CV$ALL_SEQUENCES"
- Section D.1.6, "CV$ALL_SEQUENCE_PARTITIONS"
- Section D.1.7, "CV$ALL_PUBLICATION_ITEMS_ADDED"
- Section D.1.8, "CV$ALL_PUBLICATION_ITEMS"
- Section D.1.9, "CV$ALL_PUBLICATION_ITEM_INDEXES"
- Section D.1.10, "CV.$ALL_SUBSCRIPTION_PARAMS"

D.1.1 CV$ALL_CLIENTS

The `CV$ALL_CLIENTS` view provides information about Mobile Server clients.

Table D–1 provides a description of `ALL_CLIENT` parameters.

Table D–1 ALL_CLIENTS Parameters

Column	Datatype	Null	Description
CLIENT	VARCHAR(30)	NULL	The Mobile Server client
LASTREFRESH_STARTIME	VARCHAR(19)	NULL	Start time of the last refresh session
LASTREFRESH_ENDTIME	VARCHAR(19)	NULL	End time of the last refresh session

D.1.2 CV$ALL_ERROR

The CV$ALL_ERROR view provides information about failed client transactions.

Table D–2 provides a description of ALL_ERROR parameters.

Table D–2 ALL_ERROR Parameters

Column	Datatype	Null	Description
CLIENT	VARCHAR(30)	NOT NULL	Client to which the failed transaction belongs.
TRANSACTION_ID	NUMBER(10)	NOT NULL	ID of the failed transaction.
ITEM_NAME	VARCHAR2(30)	NOT NULL	Name of the publication item that failed.
MESSAGE_TEXT	VARCHAR2(2048)	NOT NULL	Error text associated with the failed transaction and publication item.

D.1.3 CV$ALL_PUBLICATIONS

The ALL_PUBLICATIONS view provides information about Mobile Server publications.

Table D–3 provides a description of ALL_PUBLICATIONS parameters.

Table D–3 ALL_PUBLICATIONS Parameters

Column	Datatype	Null	Description
NAME	VARCHAR2(30)	NULL	Publication Name.
TYPE	VARCHAR2(40)	NULL	Publication Type.
NAME_TEMPLATE	VARCHAR2(30)	NULL	Snapshot Name Template.
ENFORCE_RI	CHAR(1)	NOT NULL	Reserved.

D.1.4 CV$ALL_SUBSCRIPTIONS

The ALL_SUBSCRIPTIONS view provides information about Mobile Server subscriptions.

Table D–4 provides a description of ALL_SUBSCRIPTION parameters.

Table D–4 ALL_SUBSCRIPTIONS Parameters

Column	Datatype	Null	Description
CLIENT	VARCHAR2(30)	NULL	The subscription's clients.
PUBLICATION	VARCHAR2(30)	NULL	The subscription's publication.
INSTANTIATED	CHAR(1)	NULL	A boolean value that indicates whether the subscription is instantiated.

D.1.5 CV$ALL_SEQUENCES

The ALL_SEQUENCES view provides information about Mobile Server sequences.

Table D–5 provides a description of ALL_SEQUENCES parameters.

Table D–5 ALL_SEQUENCES Parameters

Column	Datatype	Null	Description
NAME	VARCHAR2(30)	NULL	The sequence name.

D.1.6 CV$ALL_SEQUENCE_PARTITIONS

The ALL_SEQUENCE_PARTITIONS view provides information about Mobile Server sequence partitions.

Table D–6 provides a description of ALL_SEQUENCE_PARTITIONS parameters.

Table D–6 ALL_SEQUENCE_PARTITIONS Parameters

Column	Datatype	Null	Description
CLIENT	VARCHAR2(30)	NULL	The client to which the sequence is assigned.
NAME	VARCHAR2(30)	NULL	The sequence name.
CURR_VALUE	NUMBER(38)	NULL	The current sequence value.
INCREMENT_BY	NUMBER(38)	NULL	The sequence's increment value. The sequence increments based on this number.

D.1.7 CV$ALL_PUBLICATION_ITEMS_ADDED

The ALL_PUBLICATION_ITEMS_ADDED view provides information about Mobile Server publication items.

Table D–7 provides a description of ALL_PUBLICATION_ITEMS_ADDED parameters.

Table D–7 ALL_PUBLICATION_ITEMS_ADDED Parameters

Column	Datatype	Null	Description
PUB_NAME	VARCHAR2(30)	NULL	The publication name.
ITEM_NAME	VARCHAR2(30)	NULL	The publication item name.
OWNER	VARCHAR2(30)	NOT NULL	The base object owner.
OBJECT_NAME	VARCHAR2(30)	NOT NULL	The base object name.
TEXT	VARCHAR2(2048)	NOT NULL	The select statement.

Table D–7 (Cont.) ALL_PUBLICATION_ITEMS_ADDED Parameters

Column	Datatype	Null	Description
UPDATABLE	VARCHAR2(1)	NULL	The updatable option.
REFRESH_METHOD	CHAR(1)	NOT NULL	The refresh method. Options include fast refresh and complete refresh. Note: If you use complete refresh, it erases all of the data on the client and brings down the snapshot from the server. If your publication item is updateable, this does not cause a loss of data on the client. If you use fast refresh, then you cannot use a high priority restricting predicate.
WINNING_RULE	VARCHAR2(30)	NULL	The winning rules option for resolving replication conflicts. Options include "client wins" and "server wins".

D.1.8 CV$ALL_PUBLICATION_ITEMS

The ALL_PUBLICATION_ITEMS view provides information about Mobile Server publication items.

Table D–8 provides a description of ALL_PUBLICATION_ITEMS parameters.

Table D–8 ALL_PUBLICATION_ITEMS Parameters

Column	Datatype	Type	Description
NAME	VARCHAR2(30)	NULL	The publication item name.
OWNER	VARCHAR2(30)	NOT NULL	The owner of the publication items' base object.
OBJECT_NAME	VARCHAR2(30)	NOT NULL	Name of the base object.
TEXT	VARCHAR2(2048)	NOT NULL	The select statement.
REFRESH_METHOD	CHAR(1)	NOT NULL	The refresh method. Options include fast refresh and complete refresh. Note: If you use complete refresh, it erases all of the data on the client and brings down the snapshot from the server. If your publication item is updateable, this does not cause a loss of data on the client. If you use fast refresh, then you cannot use a high priority restricting predicate.

D.1.9 CV$ALL_PUBLICATION_ITEM_INDEXES

The ALL_PUBLICATION_ITEM_INDEXES view provides information about Mobile Server publication item indexes.

Table D–9 provides a description of ALL_PUBLICATION_ITEM_INDEXES parameters.

Table D–9 ALL_PUBLICATION_ITEM_INDEXES Parameters

Column	Datatype	Null	Description
NAME	VARCHAR2(30)	NULL	Index name.
PUB_ITEM	VARCHAR2(30)	NOT NULL	Publication item name.
INDX_TYPE	CHAR(1)	NOT NULL	Index type.
COLUMN_LIST	VARCHAR2(2048)	NOT NULL	Column list.

D.1.10 CV.$ALL_SUBSCRIPTION_PARAMS

The ALL_SUBSCRIPTION_PARAMS view provides information about Mobile Server subscription parameters.

Table D–10 provides a description of ALL_SUBSCRIPTION_PARAMS parameters.

Table D–10 ALL_SUBSCRIPTION_PARAMS Parameters

Column	Datatype	Null	Description
NAME	VARCHAR2(30)	NULL	Publication name.
CLIENT	VARCHAR2(30)	NULL	Client name.
PARAM_NAME	VARCHAR2(30)	NULL	Parameter name.
PARAM_VALUE	VARCHAR2(30)	NULL	Parameter value.

D.2 Client Oracle Lite Database System Catalogs

The following are the SQLRT and other system catalogs that are included in the client Oracle Lite database:

- Section D.2.1, "ALL_COL_COMMENTS"
- Section D.2.2, "ALL_CONSTRAINTS"
- Section D.2.3, "ALL_CONS_COLUMNS"
- Section D.2.4, "ALL_DEPENDENCIES"
- Section D.2.5, "ALL_INDEXES"
- Section D.2.6, "ALL_IND_COLUMNS"
- Section D.2.7, "ALL_OBJECTS"
- Section D.2.8, "ALL_PRIVILEGES"
- Section D.2.9, "ALL_SEQUENCES"
- Section D.2.10, "ALL_SYNONYMS"
- Section D.2.11, "ALL_TABLES"
- Section D.2.12, "ALL_TAB_COLUMNS"
- Section D.2.13, "ALL_TAB_COMMENTS"
- Section D.2.14, "ALL_USERS"
- Section D.2.15, "ALL_VIEWS"
- Section D.2.16, "POL__ALLOBJ"
- Section D.2.17, "POL__COLUSAGE"

D.2.1 ALL_COL_COMMENTS

Table D–11 ALL_COL_COMMENTS Parameters

Column	Datatype	Null	Description
OWNER	VARCHAR(128)	NOT NULL	Owner
TABLE_NAME	VARCHAR(128)	NOT NULL	Name of the table
COLUMN_NAME	VARCHAR(128)	NOT NULL	Name of the column
COMMENTS	VARCHAR(4096)	NULL	Description

D.2.2 ALL_CONSTRAINTS

Table D–12 ALL_CONSTRAINTS Parameters

Column	Datatype	Null	Description
OWNER	VARCHAR(128)	NOT NULL	Owner
CONSTRAINT_NAME	VARCHAR(128)	NOT NULL	Constraint name
CONSTRAINT_TYPE	VARCHAR(128)	NOT NULL	Constraint type
TABLE_NAME	VARCHAR(128)	NOT NULL	Name of the table
SEARCH_CONDITION	VARCHAR(1000)	NULL	Search condition
R_OWNER	VARCHAR(128)	NULL	Owner of Referenced Primary Key Constraint
R_CONSTRAINT_NAME	VARCHAR(128)	NULL	Reference Name of Primary Constraint
DELETE_RULE	VARCHAR(128)	NULL	Delete rule
STATUS	VARCHAR(20)	NOT NULL	Status
VALIDATED	VARCHAR(13)	NOT NULL	Validated

D.2.3 ALL_CONS_COLUMNS

Table D–13 ALL_CONS_COLUMNS Parameters

Column	Datatype	Null	Description
OWNER	VARCHAR(128)	NULL	Owner
CONSTRAINT_NAME	VARCHAR(128)	NULL	Constraint name
TABLE_NAME	VARCHAR(128)	NULL	Table name
COLUMN_NAME	VARCHAR(128)	NULL	Column name
POSITION	VARCHAR(10)	NULL	Position

D.2.4 ALL_DEPENDENCIES

Table D–14 ALL_DEPENDENCIES Parameters

Column	Datatype	Null	Description
OWNER	VARCHAR(30)	NOT NULL	Owner
NAME	VARCHAR(30)	NOT NULL	Name
TYPE	VARCHAR(16)	NULL	Type
REFERENCED_OWNER	VARCHAR(30)	NULL	Referenced owner
REFERENCED_NAME	VARCHAR(30)	NOT NULL	Referenced name
REFERENCED_TYPE	VARCHAR(16)	NULL	Referenced type

D.2.5 ALL_INDEXES

Table D–15 ALL_INDEXES Parameters

Column	Datatype	Null	Description
OWNER	VARCHAR(128)	NOT NULL	Owner
INDEX_NAME	VARCHAR(128)	NOT NULL	Index name
TABLE_OWNER	VARCHAR(128)	NOT NULL	Table owner
TABLE_NAME	VARCHAR(128)	NOT NULL	Table name
TABLE_TYPE	VARCHAR(10)	NULL	Table type
UNIQUENESS	VARCHAR(128)	NOT NULL	Index Uniqueness

D.2.6 ALL_IND_COLUMNS

Table D–16 ALL_IND_COLUMNS Parameters

Column	Datatype	Null	Description
INDEX_OWNER	VARCHAR(128)	NOT NULL	Index owner
INDEX_NAME	VARCHAR(128)	NOT NULL	Index name
TABLE_OWNER	VARCHAR(128)	NOT NULL	Table owner
TABLE_NAME	VARCHAR(128)	NOT NULL	Table name
COLUMN_NAME	VARCHAR(128)	NOT NULL	Column name
COLUMN_POSITION	VARCHAR(10)	NOT NULL	Column position

D.2.7 ALL_OBJECTS

Table D–17 ALL_OBJECTS Parameters

Column	Datatype	Null	Description
OWNER	VARCHAR(128)	NOT NULL	Owner
OBJECT_NAME	VARCHAR(128)	NOT NULL	Object name
OBJECT_TYPE	VARCHAR(128)	NULL	Object type
CREATED	DATE	NULL	When object was created
STATUS	VARCHAR(128)	NULL	Status

D.2.8 ALL_PRIVILEGES

Table D–18 ALL_PRIVILEGES Parameters

Column	Datatype	Null	Description
SCHEMA	VARCHAR(128)	NOT NULL	Schema
TABLE	VARCHAR(128)	NOT NULL	Table
COLUMN	NUMBER(11)	NOT NULL	Column
GRANTOR1	VARCHAR(128)	NOT NULL	Grantor
GRANTEE1	UNKNOWN(8)	NOT NULL	Grantee
OBJTYPE	TINYINT(3)	NOT NULL	Object type
GTYPE	TINYINT(3)	NOT NULL	Grant type
GTABLE	TINYINT(3)	NOT NULL	Grant table
TO	UNKNOWN(8)	NOT NULL	Not used

D.2.9 ALL_SEQUENCES

Table D–19 ALL_SEQUENCES Parameters

Column	Datatype	Null	Description
SEQUENCE_OWNER	VARCHAR(128)	NOT NULL	Sequence owner
SEQUENCE_NAME	VARCHAR(128)	NOT NULL	Sequence name
MIN_VALUE	VARCHAR(10)	NOT NULL	Minimum value
MAX_VALUE	VARCHAR(10)	NOT NULL	Maximum value
INCREMENT_BY	VARCHAR(10)	NOT NULL	Increment value for sequence

D.2.10 ALL_SYNONYMS

Table D–20 ALL_SYNONYMS Parameters

Column	Datatype	Null	Description
OWNER	VARCHAR(128)	NULL	Owner
SYNONYM_NAME	VARCHAR(128)	NULL	Synonym name

Table D–20 (Cont.) ALL_SYNONYMS Parameters

Column	Datatype	Null	Description
TABLE_OWNER	VARCHAR(128)	NULL	Table owner
TABLE_NAME	VARCHAR(128)	NULL	Table name
DB_LINK	VARCHAR(128)	NULL	Database link

D.2.11 ALL_TABLES

Table D–21 ALL_TABLES Parameters

Column	Datatype	Null	Description
OWNER	VARCHAR(128)	NOT NULL	Owner
TABLE_NAME	VARCHAR(128)	NOT NULL	Table name
TABLESPACE_NAME	VARCHAR(128)	NULL	Tablespace name
CLUSTER_NAME	VARCHAR(128)	NULL	Cluster name
PCT_FREE	NUMBER(10)	NULL	Minimum percentage of free space in a block
PCT_USED	NUMBER(10)	NULL	Minimum percentage of used space in a block. Note: If the space in data block is less than PCT_FREE, no new rows will be added in that block until amount of space in table is less than PCT_USED
INI_TRANS	NUMBER(10)	NULL	Initial number of transactions
MAX_TRANS	NUMBER(10)	NULL	Maximum number of transactions
INITIAL_EXTENT	NUMBER(10)	NULL	Set of Contiguous blocks in a database segment that is automatically allotted when a segment is created
NEXT_EXTENT	NUMBER(10)	NULL	Block allocated after initial extent
MIN_EXTENTS	NUMBER(10)	NULL	Minimum number of extents allowed in the segment
MAX_EXTENTS	NUMBER(10)	NULL	Maximum extent that can be allocated
PCT_INCREASE	NUMBER(10)	NULL	Default percentage increase in extent size
BACKED_UP	VARCHAR(1)	NULL	Has table being backed up since last modification
NUM_ROWS	NUMBER(10)	NULL	Number of rows
BLOCKS	NUMBER(10)	NULL	Number of used blocks in the table
EMPTY_BLOCKS	NUMBER(10)	NULL	Number of empty blocks in the table
AVG_SPACE	NUMBER(10)	NULL	Average available free space in the table
CHAIN_CNT	NUMBER(10)	NULL	Number of rows in the table that are chained from one data block to another or that have migrated to a new block, requiring a link to preserve the old rowid
AVG_ROW_LEN	NUMBER(10)	NULL	Average row length, including row overhead

D.2.12 ALL_TAB_COLUMNS

Table D–22 ALL_TAB_COLUMNS Parameters

Column	Datatype	Null	Description
OWNER	VARCHAR(128)	NOT NULL	Owner
TABLE_NAME	VARCHAR(128)	NOT NULL	Table name
COLUMN_NAME	VARCHAR(128)	NOT NULL	Column name
DATA_TYPE	VARCHAR(30)	NULL	Data type
DATA_LENGTH	NUMBER(10)	NULL	Data length
DATA_PRECISION	NUMBER(10)	NULL	Data precision
DATA_SCALE	NUMBER(10)	NULL	Data scale
NULLABLE	VARCHAR(1)	NULL	Nullable
COLUMN_ID	NUMBER(10)	NOT NULL	Column ID
DEFAULT_LENGTH	NUMBER(10)	NULL	Default length
DATA_DEFAULT	VARCHAR(4096)	NULL	Default value for column
NUM_DISTINCT	NUMBER(10)	NULL	Number of distinct values this column holds
LOW_VALUE	NUMBER(10)	NULL	Low value
HIGH_VALUE	NUMBER(10)	NULL	High value
IS_HIDDEN	VARCHAR(1)	NULL	If item is hidden

D.2.13 ALL_TAB_COMMENTS

Table D–23 ALL_TAB_COMMENTS Parameters

Column	Datatype	Null	Description
OWNER	VARCHAR(128)	NOT NULL	Owner
TABLE_NAME	VARCHAR(128)	NOT NULL	Table name
TABLE_TYPE	VARCHAR(128)	NOT NULL	Table type
COMMENTS	VARCHAR(4096)	NOT NULL	Description

D.2.14 ALL_USERS

Table D–24 ALL_USERS Parameters

Column	Datatype	Null	Description
USERNAME	VARCHAR(30)	NOT NULL	Username
USER_ID	NUMBER(11)	NOT NULL	User ID
CREATED	DATE	NOT NULL	When user was created

D.2.15 ALL_VIEWS

Table D–25 ALL_VIEWS Parameters

Column	Datatype	Null	Description
OWNER	VARCHAR(128)	NOT NULL	Owner
VIEW_NAME	VARCHAR(128)	NOT NULL	View name
TEXT_LENGTH	NUMBER(10)	NOT NULL	Text length
TEXT	VARCHAR(1000)	NOT NULL	Text

D.2.16 POL__ALLOBJ

Table D–26 POL__ALLOBJ Parameters

Column	Datatype	Null	Description
CATALOG_NAME	VARCHAR(128)	NOT NULL	Catalog name
SCHEMA_NAME	VARCHAR(128)	NOT NULL	Schema name
OBJECT_NAME	VARCHAR(128)	NOT NULL	Object name
OBJECT_ID	NUMBER(10)	NOT NULL	Object ID
OBJECT_TYPE	SMALLINT(5)	NOT NULL	Object type
CREATED	DATE	NOT NULL	Date when created
LAST_DDL_TIME	DATE	NOT NULL	Timestamp for the last modification of the object resulting from a DDL command

D.2.17 POL__COLUSAGE

Table D–27 POL__COLUSAGE Parameters

Column	Datatype	Null	Description
COLUMN	VARCHAR(128)	NOT NULL	Column
TBLUSAGEREF	UNKNOWN(8)	NOT NULL	Reference to its POL__TBLUSAGE object
POSITION	NUMBER(11)	NOT NULL	Column position in Constraint definition. Used to identify order of column in the key

D.2.18 POL__COMMENT

Table D–28 POL__COMMENT Parameters

Column	Datatype	Null	Description
COLREF	UNKNOWN(8)	NOT NULL	Column Reference
COMMENTS	VARCHAR(4096)	NOT NULL	Comment

D.2.19 POL__CONS

Table D–29 POL_CONS Parameters

Column	Datatype	Null	Description
CATALOG	VARCHAR(128)	NOT NULL	Catalog
SCHEMA	VARCHAR(128)	NOT NULL	Schema
CONSTRAINT	VARCHAR(128)	NOT NULL	Constraint
TYPE	NUMBER(11)	NOT NULL	Type
TBLCONS	UNKNOWN(8)	NOT NULL	Reference to POL__TBLCONS that points to constrained table
ASSERTCONS	UNKNOWN(8)	NOT NULL	Reserved
DOMAINCONS	UNKNOWN(8)	NOT NULL	Reserved
FLAGS	NUMBER(11)	NOT NULL	For update rules
TO	UNKNOWN(8)	NOT NULL	Used by foreign key constraint to point to parent table primary key constraint object
EXTENSIONS	TINYINT(3)	NOT NULL	Not used

D.2.20 POL__DATABASE_PARAMETERS

Table D–30 POL__DATABASE_PARAMETERS Parameters

Column	Datatype	Null	Description
PARAMETER	VARCHAR(30)	NOT NULL	Parameter
TYPE	TINYINT(3)	NOT NULL	Type
VALUE	VARCHAR(128)	NULL	Value

D.2.21 POL__INDICES

Table D–31 POL__INDICES Parameters

Column	Datatype	Null	Description
CATALOG_NAME	VARCHAR(128)	NOT NULL	Catalog name
SCHEMA_NAME	VARCHAR(128)	NOT NULL	Schema name
INDEX_NAME	VARCHAR(128)	NOT NULL	Index name
TABLE_SCHEMA	VARCHAR(128)	NOT NULL	Table schema
TABLE_NAME	VARCHAR(128)	NOT NULL	Table name
FLAGS	NUMBER(11)	NOT NULL	Options for Index
INDEXOBJ	UNKNOWN(8)	NOT NULL	Index object
EXTENSIONS	TINYINT(3)	NOT NULL	Not used

D.2.22 POL__INDICESDT

Table D–32 POL__INDICESDT Parameters

Column	Datatype	Null	Description
COLUMN_NAME	VARCHAR(128)	NOT NULL	Column name
DIRECTION	TINYINT(3)	NOT NULL	Ascending or descending direction
INDEX_POSITION	NUMBER(10)	NOT NULL	Relative position in index
MASTER	UNKNOWN(8)	NOT NULL	Master index table
EXTENSIONS	TINYINT(3)	NOT NULL	Not used

D.2.23 POL__PROCEDURES

Table D–33 POL__PROCEDURES Parameters

Column	Datatype	Null	Description
CATALOG_NAME	VARCHAR(128)	NOT NULL	Catalog name
SCHEMA_NAME	VARCHAR(128)	NOT NULL	Schema name
PROCEDURE_NAME	VARCHAR(128)	NOT NULL	Procedure name
TABLE_CLASS	UNKONWN(8)	NOT NULL	Table class
PROCEDURE_TYPE	SMALLINT(5)	NOT NULL	Procedure type
PROCEDURE_REF	UNKNOWN(8)	NOT NULL	Procedure reference

D.2.24 POL__PROCEDURE_COLUMNS

Table D–34 POL__PROCEDURE_COLUMNS Parameters

Column	Datatype	Null	Description
MASTER	UNKNOWN(8)	NOT NULL	Master Index table
COLUMN_TYPE	SMALLINT(5)	NOT NULL	Column type
DATA_TYPE	SMALLINT(5)	NOT NULL	Data type
TYPE_NAME	VARCHAR(128)	NOT NULL	Type name
PRECISION	NUMBER(10)	NOT NULL	Significant digits for Column
LENGTH	NUMBER(10)	NOT NULL	Length
SCALE	SMALLINT(5)	NOT NULL	Significant digits right of decimal
RADIX	SMALLINT(5)	NOT NULL	Base of the system
POSITION	SMALLINT(5)	NOT NULL	Position

D.2.25 POL__SCHEMATA

Table D–35 POL__SCHEMATA Parameters

Column	Datatype	Null	Description
NAME	VARCHAR(128)	NOT NULL	Name

Table D–35 (Cont.) POL__SCHEMATA Parameters

Column	Datatype	Null	Description
OWNER	VARCHAR(128)	NOT NULL	Owner
SCHEMA_ID	NUMBER(11)	NOT NULL	Schema ID
CREATED	DATE	NOT NULL	Date when created

D.2.26 POL__SEQ

Table D–36 POL__SEQ Parameters

Column	Datatype	Null	Description
SCHEMA_NAME	VARCHAR(128)	NOT NULL	Schema name
SEQ_NAME	VARCHAR(128)	NOT NULL	Sequence name
SEQ_REF	UNKNOWN(8)	NOT NULL	Sequence reference
INC_BY	NUMBER(10)	NOT NULL	Increment by value
START_WITH	NUMBER(10)	NOT NULL	Number to start sequence with
MIN_VALUE	NUMBER(10)	NOT NULL	Minimum value for sequence
MAX_VALUE	NUMBER(10)	NOT NULL	Maximum value for sequence
IS_ACCESSED	TINYINT(3)	NOT NULL	if accessed

D.2.27 POL__SYNONYM

Table D–37 POL__SYNONYM Parameters

Column	Datatype	Null	Description
SYN	VARCHAR(128)	NOT NULL	Synonym
OBJECT	VARCHAR(128)	NOT NULL	Object

D.2.28 POL__TBLCONS

Table D–38 POL__TBLCONS Parameters

Column	Datatype	Null	Description
CATALOG	VARCHAR(128)	NOT NULL	Fully qualified database file name
SCHEMA	VARCHAR(128)	NOT NULL	Schema
TABLE	VARCHAR(128)	NOT NULL	Table
CONSREF	UNKNOWN(8)	NOT NULL	Reference to POL__CONS object
FLAGS	NUMBER(11)	NOT NULL	Reserved
TYPE	NUMBER(11)	NOT NULL	Type
SEARCH	VARCHAR(128)	NOT NULL	Used by Check constraint to store user constraint string
SEARCHTREE	TINYINT(3)	NOT NULL	Internal representation of constraint clause

Table D–38 (Cont.) POL__TBLCONS Parameters

Column	Datatype	Null	Description
EXTENSIONS	TINYINT(3)	NOT NULL	Not used
CLASSREF	UNKNOWN(8)	NOT NULL	Reference to Class Object for the table
GROUPREF	UNKNOWN(8)	NOT NULL	Reference to Group Object for the table

D.2.29 POL__TBLUSAGE

Table D–39 POL_TBLUSAGE Parameters

Column	Datatype	Null	Description
CATALOG	VARCHAR(128)	NOT NULL	Catalog
SCHEMA	VARCHAR(128)	NOT NULL	Schema
TABLE	VARCHAR(128)	NOT NULL	Table
TBLCONSREF	UNKNOWN(8)	NOT NULL	Reference to POL__TBLCONS Object
DOMCONSREF	UNKNOWN(8)	NOT NULL	Not used
ASSERTCONSREF	UNKNOWN(8)	NOT NULL	Not used
EXTENSIONS	TINYINT(3)	NOT NULL	Not used

D.2.30 POL__TRIGGERS

Table D–40 POL__TRIGGERS Parameters

Column	Datatype	Null	Description
TRIGGER_NAME	VARCHAR(128)	NOT NULL	Trigger name
TABLE_CLASS	UNKNOWN(8)	NOT NULL	Table class
TRIGGER_KIND	SMALLINT(5)	NOT NULL	Trigger kind
METHOD	UNKNOWN(8)	NOT NULL	Method

D.2.31 POL__VIEWS

Table D–41 POL__VIEWS Parameters

Column	Datatype	Null	Description
NAME	VARCHAR(30)	NOT NULL	View name
VIEWDEF	VARCHAR(128)	NOT NULL	View definition
POSMAPS	NUMBER(11)	NOT NULL	Position Map

D.2.32 POL__USERS

Table D–42 POL__USERS Parameters

Column	Datatype	Null	Description
USERNAME	VARCHAR(30)	NOT NULL	User name

Table D–42 (Cont.) POL__USERS Parameters

Column	Datatype	Null	Description
PASSWORD	VARCHAR(30)	NOT NULL	Password
USER_ID	NUMBER(11)	NOT NULL	User ID
CREATED	DATE	NOT NULL	Date when created

E

Using the JDBC Thin Driver

To use the JDBC thin driver with Mobile Server applets, you will need to accomplish the following tasks.

1. Upload the Oracle JDBC driver file **Classes12.zip** to the Mobile Server Repository.
2. Change the applet tag of the html page to include **Classes12.zip** in the **`Archive`** tag.

The following sections describe these steps in detail. Topics include:

- Section E.1, "Upload the Oracle JDBC Driver"
- Section E.2, "Change the Applet Tag"

E.1 Upload the Oracle JDBC Driver

To upload the Oracle JDBC driver file **Classes12.zip** to the Mobile Server Repository, perform the following steps.

1. Start the `WSH` tool and enter the following statement to connect to the Mobile Server Repository.

   ```
   WSH -o mobileadmin/manager@webtogo.world
   ```

   Note: In the above statement `manager` represents the password.

2. Start the Command Prompt and enter the following.

   ```
   WEBTOGO.WORLD:/> cd webtogo

   WEBTOGO.WORLD:/webtogo>copy %ORACLE_HOME%\
   jdbc\lib\classes12.zip

   WEBTOGO.WORLD:/webtogo>exit
   ```

E.2 Change the Applet Tag

To change the applet tag, perform the following steps.

1. Change the applet tag of the HTML page to include **Classes12.zip** in the **`ARCHIVE`** tag.
2. Add the following **`ARCHIVE`** tag to all the HTML pages containing the `<APPLET>` tag.

   ```
   ARCHIVE="/webtogo/classes12.zip"
   ```

3. If the **ARCHIVE** tag already exists, you must modify it as follows.

```
ARCHIVE="/webtogo/classes12.zip, testapp.jar"
```

F

POLITE.INI Parameters

You can customize Oracle Database Lite by modifying the parameter values defined in your `POLITE.INI` file, which is available in Windows under `%WINDIR%\POLITE.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 `POLITE.INI` file.

> **Note:** On the WinCE and EPOC platforms, this file is named `POLITE.TXT`, so that you can double-click on it to open the file.

The following discusses the parameters in the different sections in the `POLITE.INI` file:

F.1 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. When you install Oracle Database Lite, the installation creates the `POLITE.INI` file in your Windows 2000, or XP home directory. On Windows CE and EPOC, the file name is `POLITE.TXT`.

The installation automatically sets the parameters in your `POLITE.INI` file, but you can modify them to customize the product behavior. To modify the `POLITE.INI` file, use an ASCII text editor.

F.2 All Databases Section

The following describes the parameters in the `[All Databases]` section of the `POLITE.INI` file.

F.2.1 CACHE_SIZE

Specifies the size of the object cache in kilobytes. The minimum is 128. If not set, the default is 4096 (4 megabytes).

F.2.2 DATA_DIRECTORY

On the WinCE platform, you may wish to define where the Oracle Lite database is installed. By default, the storage card is used—to preserve memory—and the storage card with the maximum free space is used. At least 32 MB of free space must be available. If there is not enough memory on the storage card, then the directory defaults to `\Orace`. If you want to specify the directory where the database is created, specify the directory in the `DATA_DIRECTORY` parameter, as follows:

```
DATA_DIRECTORY=\Orace
```

To synchronize, run `msync.exe`.

F.2.3 DATABASE_ID

Defines the next Database Volume ID number to be assigned the `CREATE DATABASE SQL` command. `DATABASE_ID` numbers must be unique for each database file on the system.

F.2.4 DB_CHAR_ENCODING

Specifies the Oracle Database Lite character set. If set to `NATIVE`, the default is the system default character set.

Table F–1 lists the supported code pages and their corresponding values of `DB_CHAR_ENCODING` for all supported languages.

Table F–1 Supported Code Pages and Values

Code Page	DB_CHAR_ ENCODING	Language
N/A	UTF8	All languages
(1250)	`ee8mswin1250`	(Croatian, Czech, Hungarian, Polish, Romanian, Slovak, and Slovenian)
(1251)	`c18mswin1251`	(Bulgarian, Russian, and Ukranian)
(1252)	`we8mswin1252`	(English (United States), Catalan, Danish, Dutch (Netherlands), English (United Kingdom), Finish, French (France), German (Germany), Icelandic, Italian (Italy), Malay (Malaysia), Norwegian (Bokmal), Portuguese (Brazil), Portuguese (Portugal), Spanish (Mexico), Spanish (Spain), and Swedish)
(1253)	`el8mswin1253`	(Greek)
(1254)	`tr8mswin1254`	(Turkish)
(1255)	`iw8mswin1255`	(Hebrew)
(1256)	`ar8mswin1256`	(Arabic (Egypt), and Arabic (UAE))
(1257)	`blt8mswin1257`	(Estonian and Lithuanian)
(932)	`ja16sjis`	(Japanese)
(936)	`zhs16gbk`	(Chinese (PRC) and Chinese (Singapore))
(949)	`ko16mswin949`	(Korean)
(950)	`zht16mswin950`	(Chinese (Taiwan) and Chinese (Hong Kong))

F.2.5 EXTERNAL_ENCRYPTION_DLL

You can plug-in a custom encryption module for the Oracle Lite database by adding the `EXTERNAL_ENCRYPTION_DLL` parameter to the `POLITE.INI` configuration file. Use this if you do not want to use the default AES encryption provided for the Mobile client database.

You must either implement your encryption module into a DLL for the Windows environment or into a Shared Object (.SO) for the UNIX environment.

For example, if you created the encryption module as a DLL called `my_enc.dll`, which is located in the `C:\my_dir` directory, then you would add this module as the default encryption module in the `POLITE.INI` configuration file, as follows:

```
[All Databases]
EXTERNAL_ENCRYPTION_DLL=C:\my_dir\my_enc.dll
```

For more information, see Section 16.2 "Providing Your Own Encryption Module for the Client Oracle Lite Database" in the *Oracle Database Lite Developer's Guide*.

F.2.6 FLUSH_AFTER_WRITE

Syntax

```
FLUSH_AFTER_WRITE=TRUE|FALSE
```

Default Value

```
FALSE
```

By default, the parameter FLUSH_AFTER_WRITE is disabled. Hence, writes to a database are not flushed. The last write operation during a COMMIT operation always flushes file buffers, thereby eliminating the danger of losing data. For devices that are unreliable, users can enable this flag and set the parameter to TRUE. When enabled, every write action flushes file buffers. However, this setting degrades the database COMMIT performance.

Note: This parameter applies to the WinCE platform only.

F.2.7 MAX_INDEX_COLUMNS

Defines the number of columns used in the index creation statement. For more information, see "Index Creation Options" in the *Oracle Database Lite SQL Reference*.

F.2.8 MAX_ROWS

This parameter only applies for WinCE only.

The number of rows displayed in the msql GUI tool in the tables tab. By default, this value is 20. If you want more than 20 rows displayed at a time, modify this value.

F.2.9 MESSAGE_FILE

Use the MESSAGE_FILE parameter to specify the location of the message file used for the Mobile client Oracle Lite database. The default is where the binaries are installed. You may want to modify where the message file is located if you want to test another language. Modifying the MESSAGE_FILE parameter means that you do not have to move files around to test other languages.

Configure the path and the name of the message file, as follows:

```
MESSAGE_FILE=C:\Olite\Mobile\Sdk\BIN\OLITE40.MSB
```

F.2.10 NLS_DATE_FORMAT

Allows you to use a date format other than the Oracle Database Lite default. When a literal character string appears where a date value is expected, the Oracle Database Lite tests the string to see if it matches the formats of Oracle, SQL-92, or the value specified for this parameter in the POLITE.INI file. Setting this parameter also defines the default format used in the TO_CHAR or TO_DATE functions when no other format string is supplied.

For Oracle, the default is dd-mon-yy or dd-mon-yyyy. For SQL-92, the default is yy-mm-dd or yyyy-mm-dd.

Using RR in the format forces two digit years less than or equal to 49 to be interpreted as years in the 21st century (2000–2049), and years 50 and over, as years in the 20th

century (1950–1999). Setting the RR format as the default for all two digit year entries allows you to become year-2000 compliant. For example,

```
NLS_DATE_FORMAT='RR-MM-DD'
```

You can also modify the date format using the ALTER SESSION command. For more information, see the *Oracle Database Lite SQL Reference*.

F.2.10.1 Date Format

A date format includes one or more of the elements listed in the following table. Elements that represent similar information cannot be combined, for example, you cannot use SYYYY and BC in the same format string. Table F–2 lists date formats and their corresponding description.

Table F–2 Date Formats

Format	Description
AM or P.M.	Meridian indicator, periods are optional.
PM or P.M.	Meridian indicator, periods are optional.
CC or SCC	Century, "S" prefixes BC dates with "-".
D	Day of week.
DAY	Name of day, padded with blanks to length of 9 characters.
DD	Day of month (1-31).
DDD	Day of year (1-366).
DY	Abbreviate name of day.
IW	Week of year (1-52 or 1-53) based on the ISO standard.
IYY, IY, or I	Last 3, 2, or 1 digit(s) of the ISO year, respectively.
IYYY	4-digit year, based on the ISO standard.
HH or HH12	Hour of the day (1-12).
HH24	Hour of the day (0-23).
MI	Minute (0-59).
MM	Month (01-12, for example, JAN=01).
MONTH	Name of the month, padded with blanks to length of 9 characters.
MON	Abbreviated name of the month.
Q	Quarter of the year, (1,2,3,4, for example, JAN-MAR=1).
RR	Last 2 digits of the year, for years in other countries. This forces two-digit years less than or equal to 49 to be interpreted as years in the 21st century (2000-2049), and years 50 and over, as years in the 20th century (1950-1959).
WW	Week of the year (1-53), where 1 starts on the first day of the year and continues to the seventh day of the year.
SS	Second (0-59).
SSSSS	Seconds past midnight (0-86399).
Y or YYY	Year with comma in this position.
YEAR or SYEAR	Year, spelled out. "S" prefixes BC dates with "-".

Table F–2 (Cont.) Date Formats

Format	Description
YYYY or SYYYY	4-digit year. "S" prefixes BC dates with "-".
YYY, YY, or Y	Last 3, 2, or 1 digit(s) of the year.

F.2.10.2 Date Format Examples

Listed below are sample variations of the NLS_DATE_FORMAT parameter.

1. YYYY-MONTH-DAY:HH24:MI:P.M.
2. YYYY/MONTH/DD, HH24:MI A.M.
3. YYYY-MONTH-DAY:HH24:MI:PM
4. MM D, YYY, HH:MI A.M.
5. MM, WW, RR, HH:MI A.M.
6. MM, IW, RR, HH:M1 A.M.
7. MM, DY, RR, HH:MI A.M.
8. MM; DY; IYY, HH:MI A.M.
9. MON WW, RR, HH:MI A.M.
10. MONTH.DD, SYYYY, HH:MI A.M.
11. MONTH/DD, YYYY, HH:MI A.M.
12. MONTH|DD, YYYY, HH:MI A.M.
13. MONTH DD, YYYY, HH:SSSSS:MI A.M.
14. MONTH DD, HH:SS::MI CC
15. MONTH DD, HH:SS:MI SCC
16. MONTH W, YYYY, HH:MI A.M.
17. MONTH WW, YYYY, HH:MI A.M.
18. MONTH WW, RR, HH:MI A.M.
19. MONTH WW, Q, HH:MI A.M.
20. MONTH WW, RR, HH:MI A.M.

F.2.11 NLS_LOCALE

Defines the NLS_LOCALE parameter in the POLITE.INI file to specify the locale data of Oracle Database Lite. Oracle Database Lite locale data includes the following items:

- Decimal character and group separator
- Locale currency symbol and ISO currency symbol
- Day, week, month names, and their abbreviations

For example, NLS_LOCALE=FRENCH_FRANCE specifies the locale data of FRENCH_FRANCE in Oracle Database Lite. Table F–3 describes the supported locale and corresponding values of the NLS_LOCALE setting.

Table F–3 Supported Locales and Values

Locale	NLS_LOCALE
English (United States)	AMERICAN_AMERICA
Arabic (Egypt)	ARABIC_EGYPT
Arabic (UAE)	ARABIC_UNITED ARAB EMIRATES
Bulgarian	BULGARIAN_BULGARIA
Catalan	CATALAN_CATALONIA
Chinese (PRC)	SIMPLIFIED CHINESE_CHINA
Chinese (Singapore)	SIMPLIFIED CHINESE_SINGAPORE
Chinese (Taiwan)	TRADITIONAL CHINESE_TAIWAN
Chinese (Hong Kong)	TRADITIONAL CHINESE_HONG KONG
Croatian	CROATIAN_CROATIA
Czech	CZECH_CZECH REPUBLIC
Danish	DANISH_DENMARK
Dutch (Netherlands)	DUTCH_THE NETHERLANDS
English (United Kingdom)	ENGLISH_UNITED KINGDOM
Estonian	ESTONIAN_ESTONIA
Finnish	FINNISH_FINLAND
French (France)	FRENCH_FRANCE
German (Germany)	GERMAN_GERMANY
Greek	GREEK_GREECE
Hebrew	HEBREW_ISRAEL
Hungarian	HUNGARIAN_HUNGARY
Icelandic	ICELANDIC_ICELAND
Italian (Italy)	ITALIAN_ITALY
Japanese	JAPANESE_JAPAN
Korean	KOREAN_KOREA
Lithuanian	LITHUANIAN_LITHUANIA
Malay (Malaysia)	MALAY_MALAYSIA
Norwegian (Bokmal)	NORWEGIAN_NORWAY
Polish	POLISH_POLAND
Portuguese (Brazil)	BRAZILIAN PORTUGUESE_BRAZIL
Portuguese (Portugal)	PORTUGUESE_PORTUGAL
Romanian	ROMANIAN_ROMANIA
Russian	RUSSIAN_CIS
Slovak	SLOVAK_SLOVAKIA
Slovenian	SLOVENIAN_SLOVENIA
Spanish (Mexico)	MEXICAN SPANISH_MEXICO
Spanish (Spain)	SPANISH_SPAIN

Table F–3 (Cont.) Supported Locales and Values

Locale	NLS_LOCALE
Swedish	SWEDISH_SWEDEN
Turkish	TURKISH_TURKEY
Ukrainian	UKRANIAN_UKRAINE

F.2.12 NLS_SORT

This parameter can be used to define the collation sequence for databases created on the Oracle Database Lite instance. Collation is referred as ordering strings into a culturally acceptable sequence. A collation sequence is a sequence of all collation elements from an alphabet from the smallest collation order to the largest.

```
NLS_SORT=[collation sequence]
```

When this parameter is used, all databases created with the CREATEDB command line utility or those that are replicated from the Mobile Server are enabled for the collation sequence unless a different collation sequence is specified when using the utility. Collation sequences currently supported are BINARY (default), FRENCH, GERMAN, CZECH, and XCZECH. You can only perform a linguistic sort on Oracle Lite databases that have the collation sequence of FRENCH, GERMAN, CZECH, OR XCZECH. You cannot do a linguistic sort on a BINARY collation sequence, which is used with all languages, except the three previously listed.

Note: Unless you require your databases to have linguistic sort enabled for a supported collation sequence, it is recommended that you use the CREATEDB utility with the NLS_SORT <collation sequence> parameter, which overrides this POLITE.INI parameter. Setting the NLS_SORT using the POLITE.INI file means that your databases have the specified collation sequence enabled. There is currently no way to convert a database from one collation sequence to another.

For a complete description of this feature, see Section A.2 "CREATEDB" and Section2.11 "Support for Linguistic Sort" in the *Oracle Database Lite Developer's Guide*.

F.2.13 OLITE_SERVER_LOG

The server log file contains the status of oldaemon processes including start, launch time, abort time, and executed processes. If any errors occurred, then the exception information is included. To forward all log information for a Multi-User Service on a LINUX machine, designate the filename of the logfile, as follows:

```
OLITE_SERVER_LOG = <path_and_filename>
```

F.2.14 OLITE_SERVER_TRACE

To debug the multi-user service, set this parameter to true, as follows:

```
OLITE_SERVER_TRACE = TRUE
```

See Section 2.5.1.5 "Debugging the Multi-User Service" in the *Oracle Database Lite Developer's Guide* for more information.

F.2.15 OLITE_SQL_TRACE

Generates the SQL statement text, compilation time, execution plan, and the bind value.

For example:

```
OLITE_SQL_TRACE = TRUE
```

SQL trace output is dumped to a trace file named `oldb_trc.txt` in the current working directory of the database process. For a database service on Windows, Windows NT or the Oracle Database Lite daemon for a Linux platform, the current working directory is specified by the `wdir` parameter during the database startup service or daemon. Applications that use an embedded connection to connect to the database contain a working directory. This working directory is the application working directory. To implement the tracing feature, the database process must contain permissions to create the trace file in the current working directory. The trace output is always included in the trace file. If the trace file does not exist, it is created automatically.

To modify the working directory, see Section F.2.21, "SERVICE_WDIR".

F.2.16 OLITE_WRITE_VERIFY

You can perform diagnostics if you experience database corruption due to file system write errors, I/O errors, or a media device problem. Setting `OLITE_WRITE_VERIFY` to `TRUE` generates error reporting if a checksum error occurs on the device for the Mobile client.

If you receive a `POL-3207` error, then you may wish to execute the `validatedb` tool to see if the error message came about because of a checksum error. The `validatedb` tool deciphers if a checksum error has occurred. To further diagnose the checksum error, you can set `OLITE_WRITE_VERIFY` to perform further diagnostics to see if it is a filesystem, I/O, or media problem. After you set this to true on the client, then all write operations are verified that the checksum is valid. If not, then an error is written to a log file named `<odb_file>.odb_fserr.log` in the same directory as the Oracle Lite database (ODB). At this point, only metadata is written to this log file. However, if the file has a size greater than zero, then you know that a checksum error has occurred and there is a problem on your client device.

> **Note:** Be careful in setting this parameter to `TRUE`, that you only use it while performing your diagnostic tests and that you change it back to `FALSE` when the problem is found. The error checking performed for this diagnostic effects your performance.

For example:

```
OLITE_WRITE_VERIFY = TRUE
```

F.2.17 SQLCOMPATIBILITY

Oracle Database Lite supports both Oracle SQL and SQL-92 features. For more information on Oracle SQL and SQL-92, see the *Oracle Database Lite SQL Reference*.

If there is a conflict between Oracle SQL and SQL-92, the `SQLCOMPATIBILITY` flag is referenced. If you specify `ORACLE` for the parameter, Oracle SQL is favored, and if you specify `SQL92`, SQL-92 is favored. If you do not include this parameter in the `POLITE.INI`, Oracle SQL is favored, by default.

F.2.18 TEMP_DB

The temporary database is created **by default** in virtual memory. This improves the performance of some queries that require the use of temporary tables. Unless you explicitly choose to create the temporary database in the file system with the `TEMP_DB` parameter, the `poltempx.odb` files are not created. The `*.slx` files that are sometimes used to store savepoint information are also not created. If you plan to create a large result set, you must either have enough swap space to hold the result, or choose the file option for the temporary database.

You can specify that the temporary database files are written to the file system either with the `TEMP_DB` or `TEMP_DIR` parameters. The `TEMP_DB` parameter enables you to define the name of the database files; the `TEMP_DIR` parameter allows you only to specify the directory to which the temporary database files are written.

To include this option, use the following syntax in the `POLITE.INI` file.

```
TEMP_DB=<path_and_temporary_database_name>
```

For example,

```
TEMP_DB=c:\temp\olite_
```

As a result of the example setting, Oracle Database Lite creates temporary databases as given below.

```
c:\temp\olite_0.odb, c:\temp\olite_1.odb, ...
```

F.2.19 TEMP_DIR

Specifies the directory where the temporary database `poltemp.odb` is created. If not set, the default is any `TEMP`, `TMP` or `WINDIR` setting defined in your environment. See Section F.2.18, "TEMP_DB" for more information.

F.2.20 SERVICE_PORT

Syntax

```
SERVICE_PORT=<port_number>
```

Default Value

The default port number is 1160.

Modify the default port of the multi-user service with this parameter.

F.2.21 SERVICE_WDIR

Syntax

```
SERVICE_WDIR=C:\WINDOWS\SYSTEM32
```

Modify the default working directory of the multi-user service with this parameter.

F.3 Sync Client Parameters—SYNC Section

Modify the `SYNC` section in the `POLITE.INI` file to control certain synchronization (OCAPI) functions. The following sections list the OCAPI parameters with their corresponding description and an example. OCAPI provides you with the following support functions:

- Enable the caller to start the synchronization process from the client side.
- Set flags for the synchronization session.
- Save user information locally.

> **Note:** OCAPI is only supported on the Windows 32, Windows CE, and EPOC platforms. For more information, see the *Oracle Database Lite Developer's Guide*.

F.3.1 Overview of OCAPI—msync Client API

The msync Client API (OCAPI) is a set of functions that allows programs on client devices to set synchronization parameters and start a synchronization session. You can also use this API to monitor the progress of the synchronization session. OCAPI is the interface to the client side synchronization engine.

As the Administrator, you can set the OCAPI parameters to change the default behavior of OCAPI. When you set the OCAPI parameters in the `POLITE.INI` file, then the parameter settings are implemented for the client on the first synchronization—based on the client platforms where the parameter settings need to apply.

An OCAPI function communicates with the Mobile Server through the selected transport and synchronizes the local database with the remote Mobile Server.

F.3.2 Synchronization Parameters

The following are synchronization parameters that you can modify:

F.3.2.1 TIME_LOG

Record the start and end time of a synchronization operation. OCAPI creates a table called `C$SYNC_TIME` in the `conscli.odb` file. This file logs the duration of every synchronization process. OCAPI inserts a record in the `C$SYNC_TIME` table which

stores the start and end time of every synchronization operation. The administrator can maintain a log history of synchronization times.

Example

```
TIME_LOG=TRUE
```

The above value creates a table called C$SYNC_TIME and inserts one row containing the start and end time of the synchronization process.

Default Value

FALSE

FALSE to turn off timelog feature; TRUE to enable timelog feature.

F.3.2.2 UPDATE_LOG

Set the update log file. If this parameter is set, OCAPI creates a table called C$UPDATE_LOG in the conscli.odb file. For every DML operation received from the server, OCAPI records each operation in the C$UPDATE_LOG table. Each record contains three entries namely Table Name, Client Side Row ID, and the Log Action Type. The Table Name refers to the table that the operation is performed on. The Client Side Row ID (C$UID) is a record pointer that points to the record's Row ID. Type refers to the type of DML operation such as update, insert, and delete.

Example

```
UPDATE_LOG=TRUE
```

The above value creates and inserts rows in the C$UPDATE_LOG file. FALSE to turn off update_log feature; TRUE to enable update_log feature.

Default Value

FALSE

F.3.2.3 DEBUG

View debugging messages that are sent to the debug.txt file, which includes the database name, table names, and the DML operation. When this parameter is set to 1, the debug information regarding the database name, table names, and the DML operation goes into the debug.txt file. This enables OCAPI to invoke debugging messages.

FALSE to turn off debug feature; TRUE to enable debug feature.

Default Value

FALSE

F.3.2.4 AUTO_COMMIT_COUNT

Invoke the automatic commit count feature for publication items that use manual synchronization. If this parameter is set to 0, OCAPI calls a commit count at the end of processing for each publication. If this parameter is set to 1000, OCAPI calls commits for every 1000 inserts. This value should be more than 100 and must be used only during the complete refresh process of the system.

Default Value

- 0 for Win32
- 250 for WinCE

F.3.2.5 TEMP_DIR

Specify a directory for temporary files. OCAPI creates a temporary file for saving retrieved data. When a large volume of data is being synchronized, the data received in the temporary file can be written to a flash card to save system memory. This feature is beneficial for WinCE developers. The default is the current directory (C:\). This is useful for saving memory by directing temporary files to an external storage card.

Example

```
TEMP_DIR=\Storage Card
```

OCAPI creates a temporary file on the storage card of the Windows CE application. It saves the main memory allocated for the application.

F.3.2.6 RESUME_CLIENT_TIMEOUT

The `RESUME_CLIENT_TIMEOUT` parameter is the number of seconds that the client should use to timeout network operations. The default is 60 seconds.

Set the total number of seconds that the client should use to resume network timeout operations.

Default Value

60 seconds

Example

```
RESUME_CLIENT_TIMEOUT=120
```

F.3.2.7 RESUME_CLIENT_MAXSEND

The `RESUME_CLIENT_MAXSEND` parameter is the maximum data size, in KB, that the client should send in a single POST request. This is used in cases where there is a proxy with a small limit on the data size in one request. Specifying a reasonable value, such as 256 KB, can also help clients with limited storage space, as they can free the chunks that have already been transmitted and acknowledged. The default is 1024 KB.

Set the maximum data size in KiloBytes sent by a client in a single `POST` request. Some proxies maintain fixed limits on data size in one request.

Default Value

1024

Example

```
RESUME_CLIENT_MAXSEND=2048
```

F.3.2.8 ERROR_REPORT

Set client synchronization report results for the server.

- If set to `0`, reports errors to the server during the next synchronization process.
- If set to `1`, reports errors and creates an extra connection to the server.

- If set to 2, reports synchronization success or error cases and creates an extra connection to the server.

Default Value

0

Example

```
ERROR_REPORT=2
```

F.3.2.9 DB_ENCODING

Specify client DB character encoding. This parameter value is the same as values used in Java character encoding. For more information about Java encoding, refer to the following URL:

```
http://java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html
```

This character encoding affects CHAR and VARCHAR datatypes inside client snapshot tables only.

Default Value

NULL

The default value indicates a native character set.

F.3.2.10 MEM_THRESHOLD

Set memory threshold value in bytes for synchronization. OCAPI stops synchronization operations when the available memory is less than the specified value. Under low memory conditions, applications can be unstable on a Windows CE device. OCAPI can prevent low memory conditions if you define the threshold correctly. If the available memory is lower than this value, OCAPI displays an error message.

Default Value

524288 (which is equivalent to 512KB)

F.3.2.11 VALIDATEDB

Validate the Oracle Lite database, using the `validatedb.exe` after the synchronization process. When an error is reported by the validatedb.exe, OCAPI reports the error to the server. You can set this parameter value from 0 to 100.

- If set to 100, OCAPI runs the `validatedb.exe` for every synchronization process.
- If set to 50, OCAPI runs the `validatedb.exe` for every alternate synchronization process.
- If set to 1, OCAPI runs the `validatedb.exe`, once for every 100 synchronization processes.

Default Value

0, which means that validatedb, by default, is turned off.

F.3.2.12 ENCRYPT_DB

By default, the Oracle Lite database used by the Mobile client is not encrypted. However, you can ask for it to be encrypted through the ENCRYPT_DB parameter.

EncyrptDB encrypts the Oracle Lite database by using 128 bit Advanced Encryption Standard (AES) encryption. This does not encrypt the data stored within the Oracle Lite database itself; it only encrypts the database as a whole.

> **Note:** This parameter encrypts the database using the synchronization parameter.

- If set ENCRYPT_DB to 0, encryption is not executed. The database is left in whatever current state it is in.
- If set ENCRYPT_DB to 1, encryption of the database is executed only when a new Oracle Lite database (ODB) file is created. This is the preferred method if you want an encrypted database. Thus, the database is only encrypted when it is created.
- If set ENCRYPT_DB to 2, encryption of the database runs after every synchronization process. If you already have a database that is not encrypted, then you would want to set ENCRYPT_DB to 2, perform a synchronization—after which, the database is encrypted—and then set ENCRYPT_DB back to 1. This way, the database is encrypted, but is not encrypted after every synchronization, which would be a performance hit.

EncryptDB may be executed in the following ways:

- The database may be encrypted by setting the ENCRYPT_DB=2 parameter under the SYNC category of the polite.ini file. This causes the execution of EncryptDB during the next synchronization. However, if you leave ENCRYPT_DB set to 2, it executes with every following synchronization cycle that occurs. Change the value to 1 to prevent this from executing with every synchronization cycle. If you wish to decrypt the database later on, change this to 0 and execute the DecryptDB utility.
- EncrypDB may be executed from the command line, but only for Oracle Lite database not using synchronization. The encrypdb executable is described in Section A.4 "Encrypdb" in the *Oracle Database Lite Developer's Guide*.

Even though the utility suggests that you may use EncrypDB to change the password used to connect to the device, do not attempt to use ENCRYPT_DB to change the password. This causes problems that commonly end with a Mobile client uninstall/re-install.

If the SDK version of the CAB file is used to install the Mobile Client, mSQL may also be utilized to run the EncryptDB utility. This is located by scrolling over in the tabs until the Tools section appears.

Default Value

0

F.3.2.13 SYNC_AGENT

The Synchronization Agent controls the automatic synchronization for the client. If you do not want automatic synchronization to occur at any time, then disable it by specifying No. The default is Yes.

```
[SYNC_AGENT]
```

```
ENABLE=YES|NO
```

Valid values are as follows

- YES: The Synchronization agent is enabled and can be started from the `syncagent.exe` UI. When launched from the command line, the Synchronization Agent executes as a background process

  The mSync executable starts the synchronization agent upon completion of the synchronization and if any of the client databases contain any log based snapshots.

- NO: The Synchronization agent is disabled and cannot be started from the `syncagent.exe` UI. Also, if it is launched from the command line and –start is specified, the synchronization agent terminates immediately.

  The mSync executable never starts the synchronization agent.

F.3.2.14 SSL_IGNORE_CERT

If you install the Mobile client using `setup.exe` after you create the self-signed certificate, then a message pops up asking if you want to continue. If you click Yes, then a parameter is added to the `polite.ini` that tells Oracle Database Lite to not validate the certificate. However, if you install the Mobile client using any other method, you need to set this parameter yourself. Set the `SSL_IGNORE_CERT` parameter in the `polite.ini` file to 1.

F.4 Device Management Parameters—DMC Section

This section describes parameters in the Device Management section: `DMC`. For full details on device management parameters that can be modified before installing the client, see Section 7.2, "Configuring Mobile Clients Before Installation".

The Device Management parameters are as follows:

- Section F.4.1, "DISABLE_PROMPT"
- Section F.4.2, "PUSH_PORT"
- Section F.4.3, "UPDATE_DAY and UPDATE_TIME"
- Section F.4.4, "MAX_RETRY"
- Section F.4.5, "FREQUENCY"

F.4.1 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.

F.4.2 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".

F.4.3 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 7.9, "Using the Device Manager Agent (dmagent) on the Client". If you do want to modify them here, the values are as follows:

Day when the Mobiledevice 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 Mobile 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

F.4.4 MAX_RETRY

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

F.4.5 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.

F.5 Network Parameters—NETWORK Section

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

- Section F.5.1, "DISABLE_SSL_CHECK"
- Section F.5.2, "HTTP_PROXY"

F.5.1 DISABLE_SSL_CHECK

You can use certificates that are not signed by a trusted authority on the Mobile Server. A Web-to-Go client will use any certificate for encryption without any configuration modifications. However, for all other clients, if you are using a certificate that is not signed by a trusted authority, such as a self-signed certificate, then set the following parameter in the `NETWORK` section in the `polite.ini` (`polite.txt`) file 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.

F.5.2 HTTP_PROXY

If user has a proxy between the Mobile client and 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
```

F.6 Sample POLITE.INI File

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

```
[All Databases]
DATABASE_ID=128
DB_CHAR_ENCODING=NATIVE
CACHE_SIZE=4096
MAX_INDEX_COLUMNS=5
SQLCOMPATIBILITY=SQL92
NLS_DATE_FORMAT=RR/MM/DD H24,MI,SS
NLS_LOCALE=ENGLISH
TEMP_DB=c:\temp\olite_
TEMP_DIR=D:\TMP

[SYNC]
TIME_LOG=1
UPDATE_LOG=0
```

Glossary

Connected

Connected is a generic term that refers to users, applications, or devices that are connected to a server.

Database Object

A database object is a named database structure: a table, view, sequence, index, snapshot, or synonym.

Database Server

The database server is the third tier of the Mobile Server three-tier Web model. It stores the application data.

Disconnected

Disconnected is a generic term that refers to users, applications, or devices that are not connected to a server.

Foreign Key

A foreign key is a column or group of columns in one table or view whose values provide a reference to the rows in another table or view. A foreign key generally contains a value that matches a primary key value in another table. See also "Primary Key".

Index

An index is a database object that provides fast access to individual rows in a table. You create an index to accelerate queries and sorting operations performed against the table's data. Indexes can also be used to enforce certain constraints on tables, such as unique and primary key constraints.

Indexes, once created, are automatically maintained and used for data access by the database engine whenever possible.

Integrity Constraint

An integrity constraint is a rule that restricts the values that can be entered into one or more columns of a table.

Java Applets

Java applets are small applications that are executed in the browser that extend the functionality of HTML pages by adding dynamic content.

JavaServer Pages

JavaServer Pages (JSP) is a technology that enables developers to change a page's layout without altering the page's underlying content. JSP uses HTML and pieces of Java code to combine the presentation of dynamic content with business logic.

Java Servlets

Java servlets are protocol and platform-independent server-side components that are written in Java. Java servlets dynamically extend Java-enabled servers and provide a general framework for services built using the request-response paradigm.

Java Server Development Kit

The Java Servlet Development Kit is a tool provided by Sun Microsystems for developing Java servlets.

Java Web Server Development Kit

The Java Web Server Development Kit 1.0.1 is a Sun Microsystems tool for developing both JavaServer Pages (JSP) and Java servlets.

JDBC

JDBC (Java Database Connectivity) is a standard set of Java classes providing vendor-independent access to relational data. Modeled on ODBC, the JDBC classes provide standard features such as simultaneous connections to several databases, transaction management, simple queries, manipulation of pre-compiled statements with bind variables, and calls to stored procedures. JDBC supports both static and dynamic SQL.

Join

A relationship established between keys (both primary and foreign) in two different tables or views. Joins are used to link tables that have been normalized to eliminate redundant data in a relational database. A common type of join links the primary key in one table to the foreign key in another table to establish a master-detail relationship. A join corresponds to a `WHERE` clause condition in an SQL statement.

Leapfrog Sequence

The leapfrog sequence is one of two sequence types that Web-to-Go uses in order to provide unique primary key values to the Mobile client for OC4J or Web-to-Go. Leapfrog sequences contain a different start value for each client, and each sequence increment is set to a larger value than the maximum number of clients.

Master-Detail Relationship

A master-detail relationship exists between tables or views in a database when multiple rows in one table or view (the detail table or view) are associated with a single master row in another table or view (the master table or view).

Master and detail rows are normally joined by a primary key column in the master table or view that matches a foreign key column in the detail table or view.

When you change values for the primary key, the application should query a new set of detail records, so that values in the foreign key match values in the primary key. For example, if detail records in the `EMP` table are to be kept synchronized with master records in the `DEPT` table, the primary key in `DEPT` should be `DEPTNO`, and the foreign key in `EMP` should be `DEPTNO`. See also "Primary Key" and "Foreign Key".

MIME

MIME (Multipurpose Internet Mail Extensions) is a message format used on the Internet to describe the contents of a message. MIME is used by HTTP servers to describe the type of file being delivered.

MIME Type

MIME Type is a file format defined by Multipurpose Internet Mail Extension (MIME).

Mobile client for OC4J or Web-to-Go

The Mobile Development Kit for Web-to-Go enables application developers to develop and debug Web-to-Go applications that consist of Java servlets, JavaServer Pages (JSP), or Java applets.

Mobile Manager

The Mobile Manager is a Mobile application that runs in the browser for easy administration of applications and users. Administrators use the Mobile Manager to perform such functions as granting or revoking application access to users or groups, modifying snapshot template variables, or deleting applications from the Mobile Server.

Mobile Server

The Mobile Server resides on the application server tier of the three-tier Mobile Server model and processes requests from Mobile Clients to modify data in the database server.

Mobile Server Repository

The Mobile Server Repository is a virtual file system that resides on Oracle. It is a persistent resource repository that contains all application files and definitions of the applications.

ODBC

ODBC (Open Database Connectivity) is a Microsoft standard that enables database access on different platforms. You can enable ODBC support on the Mobile client for OC4J or Web-to-Go for troubleshooting purposes. ODBC support enables you to view the client's data, which is stored on a local Oracle Lite database. To view this information, you can use SQL*Plus.

Oracle

Oracle is the database component of the Mobile Server. When the Mobile client for OC4J or Web-to-Go is connected, it stores applications and data on Oracle.

Oracle Lite

Oracle Lite is the database component of the Mobile client for OC4J or Web-to-Go. When the client is disconnected from the back-end Oracle server, it stores applications and data in the client Oracle Lite database.

Packaging Wizard

The Packaging Wizard enables developers to define and package new or existing Mobile Server applications.

Positioned Delete

A positioned `DELETE` statement deletes the current row of the cursor. Its format is as follows:

```
DELETE FROM table
   WHERE CURRENT OF cursor_name
```

Positioned Update

A positioned `UPDATE` statement updates the current row of the cursor. Its format is as follows:

```
UPDATE table SET set_list
   WHERE CURRENT OF cursor_name
```

Primary Key

A table's primary key is a column or group of columns used to uniquely identify each row in the table. The primary key provides fast access to the table's records, and is frequently used as the basis of a join between two tables or views. Only one primary key may be defined per table.

To satisfy a `PRIMARY KEY` constraint, no primary key value can appear in more than one row of the table, and no column that is part of the primary key can contain a `NULL` value.

Referential Integrity

Referential integrity is defined as the accuracy of links between tables in a master-detail relationship that is maintained when records are added, modified, or deleted.

Carefully defined master-detail relationships promote referential integrity. Constraints in your database enforce referential integrity at the database (the server in a client/server environment).

The goal of referential integrity is to prevent the creation of an orphan record, which is a detail record that has no valid link to a master record. Rules that enforce referential integrity prevent the deletion or update of a master record, or the insertion or update of a detail record, that creates an orphan record.

Registry

The registry contains a unique Web-to-Go name/value pairs. All registry names must be unique.

Replication

Replication is the process of copying and maintaining database objects in multiple databases that make up a distributed database system. Changes applied at one site are captured and stored locally before being forwarded and applied at each of the remote locations. Replication provides users with fast, local access to shared data, and protects the availability of applications because alternate data access options exist. Even if one site becomes unavailable, users can continue to query or even update the remaining locations.

Replication Conflict

Replication conflicts occur when contradictory changes to the same data are made. Replication conflicts can be avoided by proper subsetting of data. The Packaging Wizard allows the developer to specify rules on how to handle conflicts.

Schema

A schema is a named collection of database objects, including tables, views, indexes, and sequences.

Sequence

A sequence is a schema object that generates sequential numbers. After creating a sequence, you can use it to generate unique sequence numbers for transaction processing. These unique integers can include primary key values. If a transaction generates a sequence number, the sequence is incremented immediately whether you commit or roll back the transaction.

Sites

Web-to-Go creates a database for each user on the Mobile client for OC4J or Web-to-Go. This database is called a site. A client can contain multiple sites, but only one site per user. Users can have multiple sites on different clients.

Snapshots

Snapshots are copies of application data that Web-to-Go captures in real-time from the Oracle database and downloads the same to the client. A snapshot can be a copy of an entire database table, or a subset of rows from the table. When you define your snapshot, you can use the `SQL WHERE` clause to specify a parameterized SQL query, where only the row data that your application uses is downloaded to the client. Thus, you can define what is downloaded to the client: the entire contents of the table or the subset of information that is relevant to the specific user. Most applications specify a particular subset of data that is relevant only to the user to be downloaded.

Web-to-Go automatically creates the snapshots on the client machine. Each subsequent time that a user goes connects through synchronization, Web-to-Go either refreshes the snapshots with the most recent data, or recreates them depending on the complexity of the snapshot.

SQL

SQL, or Structured Query Language, is a non-procedural database access language used by most relational database engines. Statements in SQL describe operations to be performed on sets of data. When a SQL statement is sent to a database, the database engine automatically generates a procedure to perform the specified tasks.

SQL*Plus

SQL*Plus is a tool that connects to an Oracle Lite database and accesses data. You must have ODBC support enabled to use SQL*Plus.

Synchronization

Synchronization is the process Web-to-Go uses to replicate data between the Mobile client for OC4J or Web-to-Go and Oracle. Web-to-Go replicates (downloads) the user applications and data to Oracle Lite from the back-end Oracle database, based upon the SQL query defined in the publication. In addition, all modifications made on the client are uploaded to the Oracle server, if the publication is defined as updateable and not as read-only.

Synonym

A synonym is an alternative name, or alias, for a table, view, sequence, snapshot, or another synonym.

Table

A table is a database object that stores data that is organized into rows and columns. In a well designed database, each table stores information about a single topic (such as company employees or customer addresses).

Three-Tier Web Model

The three-tier Web model is an Internet database configuration that contains a client, a middle tier, and a database server. Web-to-Go architecture follows the three-tier Web model.

Transaction

A set of changes made to selected data in a relational database. Transactions are usually executed with a SQL statement such as `ADD`, `UPDATE`, or `DELETE`. A transaction is complete when it is either committed (the changes are made permanent) or rolled back (the changes are discarded).

A transaction is frequently preceded by a query, which selects specific records from the database that you want to change. See also "SQL".

Unique Key

A table's unique key is a column or group of columns that are unique in each row of a table. To satisfy a `UNIQUE KEY` constraint, no unique key value can appear in more than one row of the table. However, unlike the `PRIMARY KEY` constraint, a unique key made up of a single column can contain `NULL` values.

View

A view is a customized presentation of data selected from one or more tables (or other views). A view is like a "virtual table" that allows you to relate and combine data from multiple tables (called base tables) and views. A view is a kind of "stored query" because you can specify selection criteria for the data that the view displays.

Views, like tables, are organized into rows and columns. However, views contain no data themselves. Views allow you to treat multiple tables or views as one database object.

Web-to-Go

Oracle Web-to-Go is a framework for the creation and deployment of Mobile, Web-based, database applications. Web-to-Go contains a three-tier database architecture consisting of the Mobile client, the Mobile Server and Oracle. It is centrally managed from the server and Web-to-Go applications can be run when Web-to-Go is connected to the server or disconnected from the server. When Web-to-Go is disconnected, it stores data locally in the Cache folder and synchronizes data with the server, when it reconnects.

Mobile client for OC4J or Web-to-Go

The Mobile client for OC4J or Web-to-Go is the client tier of the Web-to-Go three-tier Web model. It contains the Mobile Server and the Oracle Lite database. Web-to-Go replicates the user applications and data to Oracle Lite database. If the publication is updateable and changes are made on the client, then Web-to-Go replicates any data changes to the back-end Oracle database.

Window Sequence

The window sequence is one of two sequences Web-to-Go uses in order to provide unique primary key values to the Mobile client for OC4J or Web-to-Go. The window sequence contains a unique range of values. The range of values never overlaps with

those of other clients. When a client uses all the values in the range of its sequence, Web-to-Go recreates the sequence with a new, unique range of values.

Workspace

The Mobile Server Workspace is a Web page that provides users with access to OC4J or Web-to-Go applications. OC4J or Web-to-Go generates the Workspace in the user's browser after the user logs in. The Workspace displays icons, links, and descriptions of all applications that are available to the user. An application is available to the user after the administrator publishes it to the Web-to-Go system and grants access privileges to the user.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z