

Oracle® Database Mobile Server

Administration and Deployment Guide

Release 12.1.0

E58650-01

January 2015

Copyright © 2014, 2015, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface	xiii
Audience	xiii
Documentation Accessibility	xiii
Related Documents	xiii
Conventions	xiii
1 Oracle Database Mobile Server Management	
1.1 Using Mobile Manager to Manage Your Mobile Server	1-1
1.1.1 Viewing Mobile Servers	1-3
1.1.1.1 Mobile Server Home Page	1-3
1.1.1.2 Manage Applications	1-5
1.1.1.3 Manage Users	1-5
1.1.1.4 Mobile Server Administration	1-5
1.1.1.5 Data Synchronization	1-6
1.1.1.6 Job Scheduler	1-6
1.1.2 Viewing Mobile Devices	1-7
1.1.2.1 Installed Mobile Devices	1-7
1.1.2.2 Mobile Device Platforms	1-7
1.2 Manage Mobile Server Farms	1-8
1.3 Enabling UIX Dynamic Image Generation on UNIX to See Mobile Manager Buttons	1-8
1.3.1 Headless Java	1-8
1.3.2 X Server Access	1-8
1.4 Mobile Server Run Time Libraries	1-8
2 Managing the Client and Back-End Databases	
2.1 Creating and Managing the Database for a Mobile Client	2-1
2.2 Connecting to the Back-End Oracle Database	2-1
3 Managing Your Mobile Applications	
3.1 Listing Applications	3-1
3.2 Publishing Applications to the Mobile Server Repository	3-2
3.3 Deleting an Application	3-2
3.4 Managing Application	3-2
3.5 Managing User-Specific Application Parameters (Data Subsetting)	3-3

3.6	Managing Access Privileges for Users and Groups.....	3-4
3.6.1	Granting Application Access to Users and Groups.....	3-4
3.6.2	Revoking Application Access to Users and Groups.....	3-5

4 Managing Users and Groups

4.1	What Are the Types of Mobile Server Users?.....	4-1
4.1.1	Mobile Server User Privilege: Administrator	4-1
4.1.2	Mobile Server User Privilege: Organizer	4-2
4.1.3	Mobile Server User Privilege: User	4-2
4.2	Guide to Creating User and Administrator Types	4-2
4.2.1	Creating a User to Access a Published Application	4-2
4.2.2	Creating an Administrator	4-3
4.3	Managing Users and Groups	4-3
4.3.1	Managing Mobile Server Users	4-3
4.3.1.1	Displaying Users.....	4-4
4.3.1.2	Adding New Users.....	4-5
4.3.1.3	Associating Mobile Server Users With Published Applications	4-7
4.3.1.4	Duplicating Existing Users.....	4-8
4.3.1.5	Swap Users on a Device.....	4-8
4.3.1.6	Managing OID Users in the Mobile Server.....	4-9
4.3.1.7	Providing Your Own Authentication for a User.....	4-10
4.3.2	Adding New Groups.....	4-11
4.3.3	Deleting Groups or Individual Users	4-11
4.4	Managing Access Privileges for Users and Groups.....	4-11
4.4.1	Grant or Revoke Application Access to Users	4-11
4.4.1.1	Grant Application Access to Users	4-12
4.4.1.2	Revoke Application Access to Users	4-12
4.4.2	Include or Exclude Users from Group Based Access	4-12
4.4.2.1	Include Users in a Group.....	4-13
4.4.2.2	Exclude Users from a Group.....	4-13
4.4.3	Grant or Revoke Application Access to Groups	4-13
4.5	Managing Application Parameter Input (Data Subsetting).....	4-14
4.6	Manually Adding Devices for a User	4-14
4.7	Configuring How the Device Receives Software Updates for the User	4-15

5 Managing Synchronization

5.1	How Does the Synchronization Process Work?	5-1
5.1.1	Defining Behavior of Apply/Compose Phase for Synchronization	5-4
5.2	Initiate Synchronization of the Mobile Client.....	5-4
5.2.1	Network Options for MSync Tool.....	5-7
5.2.2	Sync Options for MSync Tool	5-7
5.2.3	Use Mobile Client Tools on Linux.....	5-8
5.3	User Scenarios for Synchronization	5-8
5.4	Managing the Sync Server from the Data Synchronization Home Page.....	5-9
5.4.1	Starting/Stopping the Sync Server	5-9
5.4.2	Checking Synchronization Alerts.....	5-10
5.4.3	Managing Sync Sessions	5-11

5.4.4	Displaying Operating System (OS) and Java Virtual Machine (JVM) Information	5-12
5.5	Using Automatic Synchronization	5-13
5.5.1	Specifying Platform Rules for Automatic Synchronization	5-14
5.5.2	Start, Stop, or Get Status for Automatic Synchronization	5-19
5.5.3	How the Automatic Synchronization Transaction is Retried	5-20
5.6	Configuring Data Synchronization For Farm or Single Mobile Server	5-20
5.7	Resuming an Interrupted Synchronization	5-20
5.7.1	Defining Temporary Storage Location for Client Data	5-21
5.7.2	Controlling Server Load	5-21
5.8	Register a Remote Oracle Database for Application Data	5-22
5.8.1	Register or Deregister a Remote Oracle Database for Application Data	5-23
5.9	Improving Performance for Multiple Clients that Use the Same Read-Only Data With a Cached User	5-25
5.10	Synchronizing to a File With File-Based Sync	5-26
5.10.1	Upload Synchronization Transactions to an Encrypted File on the Mobile Client	5-27
5.10.2	Apply and Compose Mobile Client Transactions on the Mobile Server	5-27
5.10.3	Download Composed Transactions from Mobile Server to the Mobile Client	5-28
5.10.4	Troubleshooting File-Based Synchronization Scenarios	5-28
5.10.4.1	Normal Network Synchronization Occurs During File Upload	5-28
5.10.4.2	Conflict Resolution for File-Based Synchronization	5-29
5.11	Managing Trace Settings and Trace Files	5-29
5.12	Browsing the Repository for Synchronization Details	5-29
5.12.1	Viewing User Information	5-29
5.12.2	Viewing Publications	5-31
5.12.3	Viewing Publication Items	5-32
5.12.4	Viewing Synchronization Queues	5-32
5.12.4.1	Viewing Transactions in the In-Queue	5-32
5.12.4.2	Viewing Subscriptions in the Out-Queue	5-32
5.12.4.3	Viewing Server-Side Synchronization Conflicts and Errors in the Error Queue	5-33
5.13	Monitoring and Analyzing Performance	5-37
5.13.1	Viewing Sync Server Statistics	5-37
5.13.2	Displaying MGP Cycles and Statistics	5-38
5.13.3	Analyzing Performance of Publications With the Conspert Utility	5-39
5.13.4	Monitoring Synchronization Using SQL Scripts	5-40

6 Managing Jobs with the Job Scheduler

6.1	Scheduling a Job to Execute at a Specific Time or Interval	6-1
6.2	Managing the Job Engine	6-2
6.2.1	Starting the Job Scheduler	6-2
6.2.2	Checking Job Scheduler Alerts	6-3
6.2.3	Managing Active Jobs	6-3
6.2.4	Managing the Job History List	6-3
6.3	Manage Scheduled Jobs Using the Mobile Manager	6-4
6.3.1	Creating a New Job	6-5
6.3.2	Editing Existing Jobs	6-7
6.3.3	Viewing MGP Current Cycle Statistics	6-8

6.3.4	Viewing MGP Cycle Statistics.....	6-9
6.3.5	Enabling or Disabling Jobs	6-10
6.3.6	Deleting Jobs.....	6-10
6.3.7	Default Jobs.....	6-10
6.3.7.1	MGP_DEFAULT	6-10
6.3.7.2	PURGE_HISTORY_DEFAULT.....	6-11
6.4	Managing or Creating Jobs Using ConsolidatorManager APIs.....	6-11

7 Manage Your Devices

7.1	Customize the Mobile Client Software Installation for Your Mobile Device.....	7-1
7.2	Configuring Mobile Clients Before Installation.....	7-2
7.2.1	Modifying Device Management Parameters for Client Device	7-3
7.3	Managing Devices.....	7-5
7.3.1	Configuring the Mobile Device through Properties	7-6
7.3.1.1	Enabling or Disabling a Mobile Device.....	7-8
7.3.2	Viewing Device Information.....	7-8
7.3.3	Viewing Database Information.....	7-8
7.3.4	Viewing Software Information	7-9
7.3.5	Commands.....	7-9
7.3.6	Queue	7-9
7.3.7	Command History	7-9
7.4	Configuring and Customizing Your Mobile Device Platform	7-9
7.4.1	Modifying Platform Properties for Installation.....	7-10
7.4.2	Enabling or Disabling All Mobile Devices in a Platform.....	7-11
7.4.3	Extend or Create a Custom Platform	7-11
7.4.3.1	Enable a Platform for Your Mobile Client	7-11
7.4.3.2	Create a Custom Platform By Extending an Existing Platform	7-11
7.5	Sending Commands to Your Mobile Devices.....	7-12
7.5.1	Scheduling or Sending Commands.....	7-12
7.5.1.1	Sending Commands	7-12
7.5.1.2	Scheduling Commands.....	7-14
7.5.2	Modifying Existing Commands	7-16
7.5.2.1	Adding Parameters to Mobile Device Commands.....	7-16
7.5.3	Creating New Commands.....	7-17
7.5.4	Creating Group Commands.....	7-18
7.5.5	Enabling or Disabling Mobile Device Commands	7-18
7.5.6	Viewing the Mobile Device Command History.....	7-19
7.6	Managing Device Software Updates.....	7-19
7.6.1	Configuring the Device to Receive Required Software Updates.....	7-19
7.6.1.1	Allowing Automatic Software Updates for the Oracle Database Mobile Server Platform	7-19
7.6.1.2	Updating Application Software On Each Client.....	7-20
7.6.1.3	Rolling Out Updates With Controlled Upgrade.....	7-21
7.6.2	Configuring Application Software for Automatic Update.....	7-21
7.6.2.1	Configuring Major Software Updates for Download	7-22
7.6.2.2	Configuring Patches or Minor Updates for Download	7-22
7.6.3	Initiate Updates of Oracle Database Mobile Server Software from the Client	7-23

7.7	Using the Device Manager Agent (dmagent) on the Client	7-24
7.8	Managing the Network Protocol Between the Device and the Mobile Client Software	7-27
7.9	Installation Configuration (INF) File	7-28
7.9.1	Setup Information.....	7-29
7.9.2	Properties	7-29
7.9.3	Initialization.....	7-31
7.9.4	Including Other INF Files.....	7-31
7.9.5	INSTALL Element	7-31
7.9.5.1	DIRECTORY Section.....	7-32
7.9.5.2	FILE Section.....	7-32
7.9.5.3	ENV Section.....	7-33
7.9.5.4	REGISTRY Section.....	7-33
7.9.5.5	ODBC Section.....	7-33
7.9.5.6	LINK Section	7-34
7.9.5.7	INI Section	7-34
7.9.5.8	EXECUTE Section.....	7-35
7.9.5.9	REGISTER Section	7-35
7.9.5.10	FINISH Section.....	7-35
7.10	Defining Device Manager Commands With the Device Manager OTL Tag Language	7-36
7.10.1	Device Manager Tag Language Data Types	7-37
7.10.1.1	Character.....	7-37
7.10.1.2	Number	7-37
7.10.1.3	Integer	7-37
7.10.1.4	Long.....	7-37
7.10.1.5	Double	7-37
7.10.1.6	Boolean.....	7-37
7.10.1.7	String	7-37
7.10.1.8	Array.....	7-39
7.10.1.9	Date Methods	7-39
7.10.1.10	Time Methods	7-40
7.10.1.11	Enumeration.....	7-40
7.10.1.12	File.....	7-40
7.10.2	Operators That You Can Use With the Device Manager Tag Language.....	7-40
7.10.3	Syntax for the Device Manager Tag Language	7-41
7.10.3.1	Initialization Statements	7-41
7.10.3.2	Assignment Statements	7-41
7.10.4	Conditional Statements.....	7-42
7.10.4.1	If-Else Conditional Statement	7-42
7.10.4.2	While Conditional Statement.....	7-42
7.10.4.3	Foreach Conditional Statement	7-43
7.10.4.4	Break Statement	7-43
7.10.4.5	Choose Statement	7-43
7.10.5	Define Custom Functions	7-43
7.10.6	Manage the Database Connection.....	7-44
7.10.6.1	Specify Database Connection Information for an Application.....	7-44
7.10.6.2	Disconnect from the Database	7-44
7.10.7	Global Classes	7-44

7.10.7.1	Methods of the System Class	7-44
7.10.7.2	Methods of the DeviceManager Class	7-47
7.10.8	Importing Another OTL Page	7-49
7.10.9	Error Handling	7-49
7.10.10	Sample Device Manager Commands Using the Tag Language	7-49
7.11	Using Mobile Manager to Manage iOS Devices	7-50
7.11.1	iOS Device Enrollment	7-50
7.11.2	View iOS Device Information	7-56
7.11.3	iOS MDM Command Management	7-57
7.11.3.1	All Commands	7-57
7.11.3.2	Send Command from Mobile Manager	7-58
7.11.3.3	View Command History	7-61
7.11.3.4	Differences between iOS Device Management and Non-iOS Device Management	7-62
7.11.4	iOS Profile Management on Mobile Manager	7-62
7.11.4.1	Create or Update Profiles	7-63
7.11.4.2	Delete Profiles	7-64
7.11.4.3	Retrieve Profile Information	7-65

8 Offline Instantiation for Mobile Clients

8.1	Using Offline Instantiation to Distribute Multiple Mobile Clients	8-1
8.2	Setting Up the Mobile Server Host and Mobile Development Kit Host	8-2
8.3	Downloading the Mobile Client SETUP Executable	8-2
8.4	Configure How OLI Creates the Client Distribution Packages With the OLI Configuration File	8-3
8.4.1	SETUP	8-3
8.4.2	USERS	8-5
8.4.3	Example of OLI.INI File	8-5
8.5	Using the OLI Engine to Create and Package the Client Distribution File	8-6
8.5.1	Create and Populate Client Database Files with the MAKEDB Command	8-7
8.5.2	Package the Mobile Client Binaries with the Client Database Files with the PACKAGE Command	8-7
8.5.3	Clean Up the OLI Tables Before Executing OLI for Another Distribution	8-8
8.5.4	Check the Status of OLI Clients	8-8
8.6	Deploying Client Distribution Files on Client Machines	8-8

9 Configure Security in Oracle Database Mobile Server

9.1	Security Enhancements	9-1
9.2	Which Password is Which?	9-1
9.2.1	Modifying Repository Password	9-3
9.3	Providing Security for the Mobile Client	9-3
9.4	Configuring for Secure Socket Layer (SSL) Communication	9-4
9.4.1	Creating an SSL Certificate	9-4
9.4.2	Configuring Mobile Server for SSL	9-5
9.4.3	Troubleshooting Error Messages for an SSL-Enabled Mobile Server	9-6
9.4.4	Support for Non-SSL Mobile Clients	9-6
9.5	Providing Your Own Authentication Mechanism for Oracle Database Mobile Server ...	9-6

9.6	Using a Load Balancer.....	9-6
9.6.1	Configure the Oracle Web Cache as a Load Balancer	9-7
9.6.2	Configure Mobile Server for Load Balancer	9-7
9.6.3	Communicate with Mobile Servers.....	9-8
9.7	Using Security Manager.....	9-8
9.7.1	Enabling Java Security Manager with WebLogic Server	9-8
9.7.2	Editing Java Security Policy File.....	9-8
9.8	Using a Firewall Proxy or Reverse Proxy	9-9
9.8.1	Using a Reverse Proxy to Communicate from Internet to Intranet.....	9-9
9.8.1.1	Configure the Apache Web Server as a Reverse Proxy	9-10
9.8.1.2	Set Up Mobile Server for Mobile Client Download	9-11
9.8.1.3	Download Reverse Proxy Mobile Client.....	9-11
9.8.1.4	Enable SSL When Using a Reverse Proxy	9-12
9.8.1.5	Configure Device Management to Work With a Firewall.....	9-14
9.8.2	Using HTTP Proxy to Communicate From Inside a Firewall	9-15
9.8.2.1	Proxy Configuration for Mobile Clients.....	9-16
9.8.2.2	Proxy Configuration for the Device Manager Agent.....	9-16
9.8.2.3	Reverse Proxy Configuration for HTTP PUSH from Mobile Server Not Supported.	9-16
9.9	Providing SSL Client Authentication with a Common Access Card	9-16
9.9.1	Introduction to SSL Client Authentication	9-17
9.9.2	Smartcard and Common Access Card Overview	9-17
9.9.3	Oracle Database Mobile Server Supports Common Access Cards	9-18
9.9.4	Supported Platforms for the Common Access Card	9-18
9.9.4.1	Support for Mobile Clients That Are Not Enabled for Client Authentication .	9-18
9.9.5	Prerequisites for Common Access Card.....	9-19
9.9.6	Configuration for Client Authentication Using the Common Access Card	9-19
9.9.6.1	Configuration of the Mobile Server to Request Client Authentication	9-19
9.9.6.2	Configuration of the Mobile Client to Use a CAC.....	9-19
9.9.6.3	Configuration for Reverse Proxy and Load Balancer	9-20
9.9.7	Using the Common Access Card.....	9-20
9.10	Security Warning for Demo Applications.....	9-21

10 Reports

10.1	Viewing System Status Reports for the Server	10-1
10.2	Viewing Active User Sessions.....	10-2

11 iOS Device Management

11.1	iOS Device Management with MDM.....	11-1
11.1.1	Introduction to iOS MDM	11-1
11.1.1.1	Mobile Server and MDM.....	11-1
11.1.1.2	MDM and Push Notifications.....	11-2
11.1.2	Profiles and Device Enrollment.....	11-3
11.1.2.1	Configuration Profiles.....	11-3
11.1.2.2	Device Enrollment Process.....	11-6
11.1.3	iOS MDM Commands.....	11-6

11.1.3.1	Command Status	11-6
11.1.3.2	Commands.....	11-8
11.1.4	Managing iOS Applications	11-10
11.1.4.1	Provisioning Profiles.....	11-10
11.1.4.2	Preparing and Publishing iOS Application	11-11
11.1.4.3	Application Security Considerations.....	11-16
11.2	Server and Security Configuration for iOS Mobile Device Management	11-17
11.2.1	Creating and Configuring Certificate Authority	11-17
11.2.1.1	Creating Private CA	11-18
11.2.1.2	Importing the CA into Mobile Server MDM Keystore	11-19
11.2.1.3	Importing the CA into the Application Server Trust Store	11-20
11.2.1.4	Important Considerations	11-22
11.2.2	SSL Configuration for MDM.....	11-23
11.2.2.1	Creating SSL Certificate Signed by Private CA.....	11-23
11.2.2.2	Importing SSL Certificate into Application Server's Keystore	11-24
11.2.3	Obtaining and Importing APNS Certificate	11-26
11.2.3.1	Obtain MDM CSR Signing Certificate.....	11-27
11.2.3.2	Create APNS CSR.....	11-28
11.2.3.3	Sign CSR with MDM CSR Signing Certificate	11-28
11.2.3.4	Submit CSR and Obtain APNS Certificate.....	11-29
11.2.3.5	Import APNS Certificate and Private Key into Mobile Server MDM Keystore	11-29
11.2.3.6	Renewing MDM Certificates.....	11-30
11.2.3.7	Important Considerations	11-31

A Configuration Files for the Mobile Server

A.1	Configuration Parameters for the MOBILE.ORA File.....	A-1
A.1.1	[APPLICATIONS].....	A-1
A.1.2	[MOBILE]	A-1
A.1.3	[FILESYSTEM].....	A-3
A.1.4	[DEBUG]	A-4
A.1.5	[CONSOLIDATOR].....	A-6
A.1.5.1	Data Synchronization Parameters.....	A-6
A.1.5.2	Data Synchronization Tracing and Logging	A-10
A.1.6	[EXTERNAL_AUTHENTICATION]	A-13
A.2	Data Synchronization Requirements in INIT.ORA.....	A-13
A.2.1	Relationships Between Relevant Parameters.....	A-13
A.2.2	Values for Processes and DML Locks.....	A-13

B Write Scripts for the Mobile Server With the WSH Tool

B.1	Description of Syntax for WSH Batch Scripts	B-1
B.1.1	Creating a User.....	B-1
B.1.1.1	EXTERNALUSER Parameter	B-2
B.1.1.2	PRIVILEGE Parameter	B-2
B.1.2	Creating a Group	B-2
B.1.3	Adding Users to a Group	B-2
B.1.4	Removing Users from a Group.....	B-2

B.1.5	Creating Access Privileges	B-3
B.1.6	Granting Access	B-3
B.1.7	Revoking Access	B-3
B.1.8	Creating Registries.....	B-3
B.1.9	Creating Snapshot Variables.....	B-4
B.1.10	Deleting a User.....	B-4
B.1.11	Deleting a Group.....	B-4
B.1.12	Deleting Access Privileges.....	B-4
B.1.13	Deleting a Registry	B-4
B.1.14	Deleting Snapshot Variables	B-4
B.2	Running a Script INI File With the WSH Tool	B-5
B.3	Examples of Batch Script Files for WSH.....	B-5
B.3.1	Creating, Adding, and Granting Access	B-5
B.3.2	Deleting, Removing, and Revoking Access	B-7

C Catalog Views for the Mobile Server

C.1	CV\$ALL_CLIENTS.....	C-1
C.2	CV\$ALL_ERRORS.....	C-1
C.3	CV\$ALL_PUBLICATIONS.....	C-2
C.4	CV\$ALL_SUBSCRIPTIONS	C-2
C.5	CV\$ALL_SEQUENCES.....	C-2
C.6	CV\$ALL_SEQUENCE_PARTITIONS.....	C-3
C.7	CV\$ALL_PUBLICATION_ITEMS_ADDED.....	C-3
C.8	CV\$ALL_PUBLICATION_ITEMS.....	C-4
C.9	CV\$ALL_PUBLICATION_INDEXES	C-4
C.10	CV\$ALL_SUBSCRIPTION_PARAMS	C-4

D iOS Mobile Device Management (MDM) Architecture

D.1	Description - MDM Architecture.....	D-1
D.1.1	MDM Command Cycle.....	D-1
D.1.2	Additional Resources	D-3
D.2	Device Enrollment Process	D-3
D.3	Application Server Configuration for MDM	D-5
D.3.1	Glassfish	D-5
D.3.2	WebLogic	D-7

Glossary

Index

Preface

This preface introduces you to the *Oracle Database Mobile Server 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

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

Access to Oracle Support

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

Related Documents

Use the following manuals as reference when performing administration tasks for either a WebLogic or Glassfish server:

- *Oracle® Fusion Middleware documentation for Oracle WebLogic Server*
- *Oracle® GlassFish Server 3.1 documentation*

Conventions

The following conventions are also used in this manual:

Convention	Meaning
.	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.

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

Oracle Database Mobile Server Management

The following sections describe management for Oracle Database Mobile Server by using the Mobile Manager for managing the mobile server and its mobile clients.

- [Section 1.1, "Using Mobile Manager to Manage Your Mobile Server"](#)
- [Section 1.2, "Manage Mobile Server Farms"](#)
- [Section 1.3, "Enabling UNIX Dynamic Image Generation on UNIX to See Mobile Manager Buttons"](#)
- [Section 1.4, "Mobile Server Run Time Libraries"](#)

1.1 Using Mobile Manager to Manage Your Mobile Server

The Mobile Manager only displays the relevant functionality for the user who logs in. Use the Mobile Manager to manage mobile applications and its users.

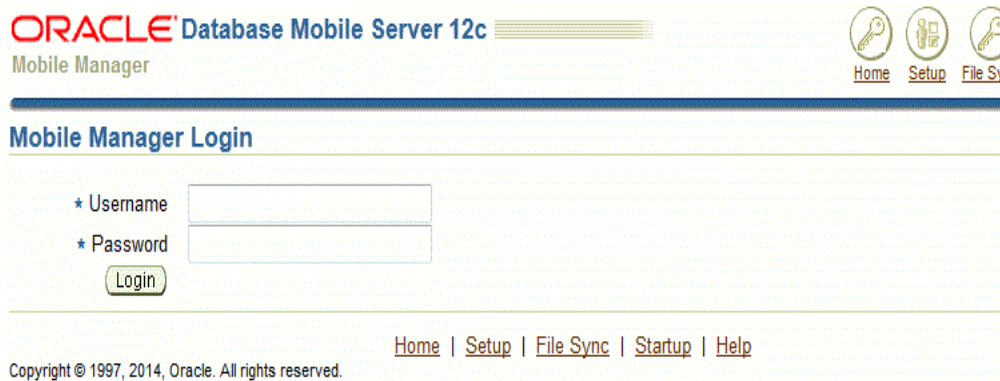
Note: See the *Oracle Database Mobile Server Mobile Client Guide* for more information on how to manage and synchronize the mobile client.

The 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. Connect to the Mobile Server (using a browser) by entering the following Mobile Server URL:

```
http://<Mobile_Server_hostname>:<Mobile_Server_port>/mobile
```

[Figure 1–1](#) displays the Mobile Manager Login page.

Figure 1–1 Mobile Manager Logon Page


ORACLE Database Mobile Server 12c
Mobile Manager

Home Setup File Sync

Mobile Manager Login

* Username

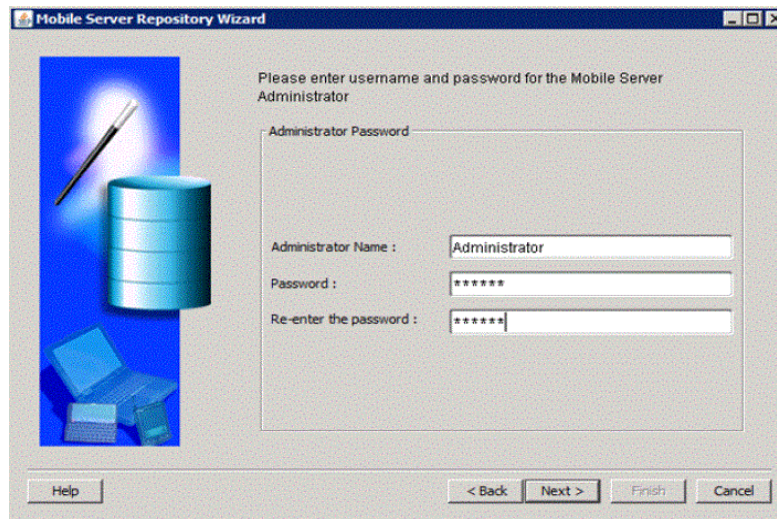
* Password

Login

Home | Setup | File Sync | Startup | Help

Copyright © 1997, 2014, Oracle. All rights reserved.

2. Log on to the Mobile Manager with the Mobile Server administrator user name and password, which were specified during the repository setup step of the Mobile Server installation process. See [Figure 1–2](#).

Figure 1–2 Mobile Server Administrator


Mobile Server Repository Wizard

Please enter username and password for the Mobile Server Administrator

Administrator Password

Administrator Name : Administrator

Password : *****

Re-enter the password : *****

Help < Back Next > Finish Cancel

Note: See [Section 4.3.1.2.1, "Define User Name and Password"](#) for conventions for creating the user name or password.

3. As [Figure 1–3](#) displays, the mobile server farm page appears with a list of installed mobile servers that use the same repository.

The mobile servers farm page 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, Mobile Server Up or Down Status, the MGP Up or Down, Start Time of the Mobile Server instance, and mobile server version.

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 the appropriate 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.

Figure 1–3 Mobile Server Farm Page

ORACLE Database Mobile Server 12c
Mobile Manager

Mobile Servers | Mobile Devices

Mobile Servers

Page Refreshed Sep 17, 2014 2:29:07 PM

Search

Host Name	Port	SSL	Status	MGP	Start Time	Version
bdbcn6.cn.oracle.com	9091				Sep 16, 2014 10:05:21 AM	12.1.0.0.0

[Logout](#) | [Help](#)

Copyright © 1997, 2014, Oracle. All rights reserved.

If you are using a browser from a different machine than the Mobile Server to access the Mobile Manager, then those machines must be using compatible character sets.

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:

- [Section 1.1.1, "Viewing Mobile Servers"](#)
- [Section 1.1.2, "Viewing Mobile Devices"](#)

1.1.1 Viewing Mobile Servers

From the mobile server farm page, select the mobile server on which you want to administer. This displays the home page for that mobile server.

The following sections describe the mobile server pages and how you can use them to administer your mobile server and its components:

- [Section 1.1.1.1, "Mobile Server Home Page"](#)
- [Section 1.1.1.2, "Manage Applications"](#)
- [Section 1.1.1.3, "Manage Users"](#)
- [Section 1.1.1.4, "Mobile Server Administration"](#)
- [Section 1.1.1.5, "Data Synchronization"](#)
- [Section 1.1.1.6, "Job Scheduler"](#)

1.1.1.1 Mobile Server Home Page

The mobile server home page provides a reference to the working status of its components. It displays if the mobile server is running or shut down and properties of the database in which the repository resides. When required to lookup current MGP Cycles, In Queue transactions, Out Queue publications, and transactions listed in the Error Queue, use the given links. Alerts are displayed as exceptions and failures for the Synchronization server, MGP, and the Job Scheduler. You use the mobile server

home page to start or shut down the Synchronization server and the Job Scheduler. You also access these components from this page.

In addition, from this page, you can navigate to display mobile server applications, users, and other administrative details by selecting the Applications, Users, or Administration tabs.

The following sections describe information displayed on the mobile server home page.

- [Section 1.1.1.1.1, "Checking Mobile Server Status"](#)
- [Section 1.1.1.1.2, "Viewing Main Database Properties"](#)
- [Section 1.1.1.1.3, "Monitoring Data Synchronization"](#)
- [Section 1.1.1.1.4, "Tracking Synchronization, MGP, and Job Scheduler Alerts"](#)
- [Section 1.1.1.1.5, "Starting or Stopping Mobile Server Components"](#)

1.1.1.1.1 Checking Mobile Server Status General information such as current mobile server status, version, and mode. To lookup if the mobile server is running or not, you refer to this section. It also displays general information about the mobile server such as current status, version, date and time since it has been running, and installation mode. This information can be used when reporting a problem to Oracle Support.

1.1.1.1.2 Viewing Main Database Properties Displays properties of the main database in which the mobile server repository resides, such as database version, JDBC URL, JDBC Driver, JDBC version, and schema name.

1.1.1.1.3 Monitoring Data Synchronization Data Synchronization information such as MGP Job status, In Queue, Out Queue, and Error Queue details. See [Chapter 5, "Managing Synchronization"](#) for more information on Data Synchronization and the queues.

- To check the status of any of the MGP Job processes, click the Status link that is displayed for the desired MGP Cycle Status. The MGP Cycle page for that MGP Job displays the MGP cycle summary, lists the log tables that have been processed and corresponding dirty record count, and lists MGP Cycle statistics for users.
- The "In Queue" link displays the number of transactions that are currently present in the In Queue. To lookup these transactions, click the "In Queue" link. The "In Queue" transactions page allows you to search for transactions by user and lists current transactions in the In Queue.
- The "Out Queue" link displays the number of publications that are currently present in the Out Queue. To lookup these publications, click the "Out Queue" link. The "Out Queue" page allows you to search for publications by user and lists current publications in the Out Queue. You can also select an Out Queue publication and display the publication items that are associated with it.
- The "Error Queue" link displays the number of transactions that are currently present in the Out Queue. To lookup these transactions, click the "Error Queue" link. The "Error Queue" page also allows you to search for transactions that contain errors by user and lists current transactions in the Error Queue.

1.1.1.1.4 Tracking Synchronization, MGP, and Job Scheduler Alerts Alert details that describe alert severity and the date and time on which the alert was triggered. Based on the alert severity and the date and time on which the alert was triggered, information displayed in the Alerts table enables you to drill down to the root cause of an error and resolve it accordingly.

Your mobile server may encounter the following alerts.

- Synchronization server exceptions
- User synchronization failures
- Job scheduler exceptions
- Job failures
- MGP Job exceptions
- MGP User Apply or Compose failures

To view alert details, select the "Alert Name" and click "Check".

1.1.1.1.5 Starting or Stopping Mobile Server Components The mobile server home page provides controls to start and shut down the Synchronization server and the Job Scheduler. It displays the current status of these components and time since they are operating in the current status.

By default, the Data Synchronization server and Job Scheduler are available is running mode. To stop the Data Synchronization server or the Job Scheduler for maintenance or debugging, select the required component and click "Stop". The Job Scheduler stops after scheduled jobs have been executed and synchronization sessions have been completed. To restart, select the required component and click "Start".

See [Chapter 5, "Managing Synchronization"](#) for more information on Data Synchronization. See [Chapter 6, "Managing Jobs with the Job Scheduler"](#) for more information on the Job Scheduler.

1.1.1.2 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.

See [Chapter 3, "Managing Your Mobile Applications"](#) for more information on how to manage your applications.

1.1.1.3 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.

1.1.1.4 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

Mobile Server

[Home](#)
[Applications](#)
[Users](#)
[Administration](#)

Page Refreshed Aug 15, 2011 8:51:01 AM

[Sessions](#)
[Trace Settings](#)
[Edit Config file](#)
[Summary](#)

[Home](#)
[Applications](#)
[Users](#)
[Administration](#)

Components

Select	Name	Status	Current Status Since	Up Time (days)	Active Sessions/Jobs
<input checked="" type="radio"/>	Data Synchronization		Aug 12, 2011 3:30:05 PM	2.72	0
<input type="radio"/>	Job Scheduler		Aug 12, 2011 3:30:05 PM	2.72	0

1. View the sessions that are active. See [Section 10.2, "Viewing Active User Sessions"](#) for full details.
2. Edit trace settings. See Chapter 3, "Tracing and Logging" in the *Oracle Database Mobile Server 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 `mobile.ora` file directly, you can access it with this link. See [Appendix A, "Configuration Files for the Mobile Server"](#) for details on the parameters.
4. View a summary of the database, JDK, and Operating System. See [Section 10.1, "Viewing System Status Reports for the Server"](#) for full details.
5. Upload an SSL certificate. When you are using an SSL connection with a reverse proxy, you need to upload an SSL certificate. See [Section 9.8.1.4, "Enable SSL When Using a Reverse Proxy"](#) for full details.

1.1.1.5 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 occurring.
- View statistics of previously executed synchronization sessions.
- Execute the Conspert 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.

1.1.1.6 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.

1.1.2 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 1.1.2.1, "Installed Mobile Devices"](#)
- [Section 1.1.2.2, "Mobile Device Platforms"](#)

For full details on managing your mobile devices, see [Chapter 7, "Manage Your Devices"](#).

1.1.2.1 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*

The screenshot shows the Oracle Database Mobile Server 12c Mobile Manager interface. The page title is "Mobile Servers | Mobile Devices". Below the title, there are tabs for "Devices", "Platforms", and "Administration". The "Devices" tab is selected. The page displays a table with the following data:

Select	Device Name	Enabled	Owner	Platform	Version	Last Accessed
<input type="checkbox"/>	WIHU-CN2.cn.oracle.com-x86	<input checked="" type="checkbox"/>	JOHN	BDB WIN32	12.1.0.0.0	Sep 17, 2014 2:42:46 PM

The page also includes a search bar, a "Page Refreshed" timestamp of "Sep 17, 2014 3:04:41 PM", and a footer with the text "Copyright © 1997, 2014, Oracle. All rights reserved." and "Logout | Help" links.

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"](#).

1.1.2.2 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 Mobile Server 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"](#).

1.2 Manage Mobile Server Farms

When you configure multiple mobile servers against a single repository, this is known as a farm.

To enable the Device Manager, Mobile Manager and all of the mobile clients to work properly in Farms, you need to modify the `mobile.ora` file on EACH mobile server.

Add and enable the following parameter in the `[MOBILE]` section of `mobile.ora` file on all mobile servers in the farm:

```
DM_AUTO_SYNC_CACHE=YES
```

Then, to enable synchronization for all mobile servers in the farm, perform the tasks described in [Section 5.6, "Configuring Data Synchronization For Farm or Single Mobile Server"](#).

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

In order to avoid X server configuration issues, enable headless operation by setting the `java.awt.headless` Java option to `true`.

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>
```

1.4 Mobile Server Run Time Libraries

You can find mobile server run time libraries in the following folders after you finish database mobile server installation. They are located at different locations for mobile servers on different application servers.

For mobile server on top of Glassfish or Weblogic or OracleAS, those run time libraries are in `$MOBILE_HOME\mobile\server\admin\repository\mobile\lib` and `$MOBILE_HOME\mobile\server\admin\repository\mobile\console\WEB-INF\lib`, where `$MOBILE_HOME` is the installation directory of mobile server.

For mobile server on top of TomEE, those run time libraries are in `$TOMEE_HOME\apps\mobile\lib` and `$TOMEE_HOME\apps\mobile\console\WEB-INF\lib`, where `$TOMEE_HOME` is TomEE application server installation directory.

Managing the Client and Back-End Databases

The following sections describe how to create and manage the client database and how to connect to the back-end Oracle database:

- [Section 2.1, "Creating and Managing the Database for a Mobile Client"](#)
- [Section 2.2, "Connecting to the Back-End Oracle Database"](#)

2.1 Creating and Managing the Database for a Mobile Client

When you use the mobile client and mobile server to replicate data between the back-end Oracle database and your mobile device, a small database is created on your mobile device to contain the data—that is stored in tables known as snapshots. The snapshot tables are used to track the modifications that the client makes on the data, which is then replicated during the synchronization process to the back-end database. All of this activity is transparent to the client. Your application queries and modifies data using SQL as if interacting with any Oracle database.

The client database is automatically created on the first synchronization request. In addition, the data is replicated and updated with the data on the Oracle database automatically for you. See Section 2.3, "What is the Process for Setting Up a User for Synchronization?" in the *Oracle Database Mobile Server Developer's Guide* for techniques that can be used to create publication items on the mobile server, which then automatically creates snapshots on the client when you synchronize with the database.

2.2 Connecting to the Back-End Oracle Database

When you are connecting to the back-end Oracle database, use the JDBC driver. When connecting to the repository in the back-end Oracle database, provide a URL to locate the database. Use the Thin JDBC driver to connect to the back-end Oracle database. With the Thin JDBC driver, you can either provide the host, port and SID, the Oracle Net address or tnsnames entry.

For oracle database 11g or below, the syntax for providing the host, port and SID is as follows:

```
jdbc:oracle:thin:@<hostname>:<port>:<sid>
```

For example, the following JDBC URL for the thin JDBC driver connects to my-pc.us.oracle.com on port 1521 with SID of orcl:

```
jdbc:oracle:thin:@my-pc.us.oracle.com:1521:orcl
```

For oracle database 12c, the syntax for providing the host, port and service name is as follows:

```
jdbc:oracle:thin://<hostname>:<port>/<service name>
```

For example, the following JDBC URL for the thin JDBC driver connects to my-pc.us.oracle.com on port 1522 with service name of pdborcl.us.oracle.com:

```
jdbc:oracle:thin://my-pc.us.oracle.com:1522/pdborcl.us.oracle.com
```

The syntax for using an Oracle Net address or tnsnames entry is as follows:

```
jdbc:oracle:thin:@oracle-net-address-or-tnsnames-entry
```

For example, the following JDBC URL for the thin JDBC driver connects to the MyDB tnsnames entry:

```
jdbc:oracle:thin:@MyDb
```

Alternatively, you could provide the full Oracle Net address, as follows:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)
(HOST=my-pc.us.oracle.com) (PORT=1521))) (CONNECT_DATA=(SERVICE_NAME=mypc)))
```

If the mobile server repository is installed in an Oracle RAC database, then provide the JDBC URL for an Oracle RAC database, which can have more than one address for multiple Oracle databases in the cluster. The following is the URL structure for an Oracle RAC database address:

```
jdbc:oracle:thin:@(DESCRIPTION=
(ADDRESS_LIST=
(ADDRESS=(PROTOCOL=TCP) (HOST=PRIMARY_NODE_HOSTNAME) (PORT=1521))
(ADDRESS=(PROTOCOL=TCP) (HOST=SECONDARY_NODE_HOSTNAME) (PORT=1521))
)
(CONNECT_DATA=(SERVICE_NAME=DATABASE_SERVICENAME)))
```


Managing Your Mobile Applications

The administrator manages applications through the following tasks:

- Section 3.1, "Listing Applications"
- Section 3.2, "Publishing Applications to the Mobile Server Repository"
- Section 3.3, "Deleting an Application"
- Section 3.4, "Managing Application"
- Section 3.5, "Managing User-Specific Application Parameters (Data Subsetting)"
- Section 3.6, "Managing Access Privileges for Users and Groups"

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

Mobile Server

[Home](#) [Applications](#) [Users](#) [Administration](#)

Page Refreshed Aug 15, 2011 9:00:28 AM

Search

Select	Application Name <small>▲</small>	Status	Virtual Path	Platform
<input checked="" type="radio"/>	Transport_PPC60.ARMV4I	✔	/Transport_PPC60.ARMV4I	
<input type="radio"/>	Transport_WIN32	✔	/Transport_WIN32	

[Home](#) [Applications](#) [Users](#) [Administration](#)

Components

Select	Name	Status	Current Status Since	Up Time (days)	Active Sessions/Jobs
<input checked="" type="radio"/>	Data Synchronization	✔	Aug 12, 2011 3:30:05 PM	2.72	0
<input type="radio"/>	Job Scheduler	✔	Aug 12, 2011 3:30:05 PM	2.72	0

To search applications, enter your application name in the "Search" field and click "Go". The Applications page displays the search result.

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 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.

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 assigns 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 Oracle Database 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 synchronization fails.

You should never delete an application if you have existing mobile clients that still need to synchronize with this application.

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 Managing Application

From the "Applications" page, you can modify application 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.

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

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 4.5, "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 4.4.1, "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 describes how an administrator may grant or revoke application access to users and groups:

- [Section 3.6.1, "Granting Application Access to Users and Groups"](#)
- [Section 3.6.2, "Revoking Application Access to Users and Groups"](#)

3.6.1 Granting 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 `Transport_Win32` application. Select the users or groups that you want `Transport_Win32` to have access to and click "Save". In this example, the administrator granted access for the `Transport_Win32` 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

Application: Transport_WIN32

[Properties](#) | [Access](#) | [Data Subsetting](#) | [Files](#)

Page Refreshed Aug 15, 2011 9:05:58 AM

Groups

[Select All](#) | [Select None](#)

Select	Group Name ^	Roles
<input type="checkbox"/>	PUBLIC GROUP	
<input type="checkbox"/>	SAMPLE USERS	

Users

[Select All](#) | [Select None](#)

Select	User Name ^	Display Name	Roles
<input type="checkbox"/>	AA	aa	
<input checked="" type="checkbox"/>	JACK	Sample User Jack	
<input checked="" type="checkbox"/>	JANE	Sample User Jane	
<input checked="" type="checkbox"/>	JOHN	Sample User John	
<input type="checkbox"/>	JUNE	Sample User June	
<input type="checkbox"/>	S11U1	S11U1	

[Properties](#) | [Access](#) | [Data Subsetting](#) | [Files](#)

3.6.2 Revoking 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".

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:

- [Section 4.1, "What Are the Types of Mobile Server Users?"](#)
- [Section 4.2, "Guide to Creating User and Administrator Types"](#)
- [Section 4.3, "Managing Users and Groups"](#)
- [Section 4.4, "Managing Access Privileges for Users and Groups"](#)
- [Section 4.5, "Managing Application Parameter Input \(Data Subsetting\)"](#)
- [Section 4.6, "Manually Adding Devices for a User"](#)
- [Section 4.7, "Configuring How the Device Receives Software Updates for the User"](#)

4.1 What Are the Types of Mobile Server Users?

The 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.

- [Section 4.1.1, "Mobile Server User Privilege: Administrator"](#)
- [Section 4.1.2, "Mobile Server User Privilege: Organizer"](#)
- [Section 4.1.3, "Mobile Server User Privilege: User"](#)

4.1.1 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 4.1.3, "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 4.3.1.3, "Associating Mobile Server Users With Published Applications"](#) for more information on this process.

4.1.2 Mobile Server User Privilege: Organizer

The organizer can perform the following tasks.

- The organizer user can use organizer as the user name and password when logging in to a mobile server application on a device.
- The organizer can publish applications through the Packaging Wizard only. A user with this privilege cannot log in to the Mobile Manager and perform administration tasks.

4.1.3 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 user name/password for synchronizing the application from a device.

Note: See [Section 4.3.1.2.1, "Define User Name and Password"](#) for conventions for creating the user name or password.

Thus, this mobile server user enables access to a particular mobile application and its publication items. That is, in order for the Windows Mobile or other devices to be able to synchronize and retrieve a snapshot of data from the database, the mobile server validates that the user name/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 4.3.1.3, "Associating Mobile Server Users With Published Applications"](#) for more information.

Note: You can swap out users for a single device. See [Section 4.3.1.5, "Swap Users on a Device"](#) for more information.

4.2 Guide to Creating User and Administrator Types

The following sections provide an overview of how to create all user types:

- [Section 4.2.1, "Creating a User to Access a Published Application"](#)
- [Section 4.2.2, "Creating an Administrator"](#)

4.2.1 Creating a User to Access a Published Application

To create any user, including administrators, to access published applications, perform 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 4.3.1.2, "Adding New Users"](#) for more information.
2. Associate the new user with the application as described in [Section 4.3.1.3, "Associating Mobile Server Users With Published Applications"](#).
3. Associate the users or groups with the application. See [Section 4.4, "Managing Access Privileges for Users and Groups"](#) for more information.

Note: You can share a device among several users by swapping out users on that device. For more information, see [Section 4.3.1.5, "Swap Users on a Device"](#).

4. 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 4.5, "Managing Application Parameter Input \(Data Subsetting\)"](#) for more information.

You now have a new user or group that is associated with an application.

4.2.2 Creating an Administrator

In order to log in as an administrator with a user name/password that is different from the administrator created upon installation, perform the following:

1. As described in [Section 4.2.1, "Creating a User to Access a Published Application"](#), 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.

4.3 Managing Users and Groups

The following sections discuss how to manage users:

- [Section 4.3.1, "Managing Mobile Server Users"](#)
- [Section 4.3.2, "Adding New Groups"](#)
- [Section 4.3.3, "Deleting Groups or Individual Users"](#)

4.3.1 Managing Mobile Server Users

The following sections define the user types and describe how to manage your users:

- [Section 4.3.1.1, "Displaying Users"](#)
- [Section 4.3.1.2, "Adding New Users"](#)
- [Section 4.3.1.3, "Associating Mobile Server Users With Published Applications"](#)
- [Section 4.3.1.4, "Duplicating Existing Users"](#)
- [Section 4.3.1.5, "Swap Users on a Device"](#)
- [Section 4.3.1.6, "Managing OID Users in the Mobile Server"](#)

- [Section 4.3.1.7, "Providing Your Own Authentication for a User"](#)

4.3.1.1 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. Refer to [Figure 1–3](#), for the mobile servers farm page.

Click your mobile server name link. Your mobile server home page appears. Click the "Users" link. As [Figure 4–1](#) displays, the Users page lists existing groups and individual users.

Figure 4–1 Users Page

Groups

Search

[Select All](#) | [Select None](#)

Select	Group Name
<input type="checkbox"/>	PUBLIC GROUP
<input type="checkbox"/>	SAMPLE USERS

Users

Search

[Select All](#) | [Select None](#)

Select	User Name	Display Name
<input type="checkbox"/>	AA	aa
<input type="checkbox"/>	JACK	Sample User Jack
<input type="checkbox"/>	JANE	Sample User Jane
<input type="checkbox"/>	JOHN	Sample User John
<input type="checkbox"/>	JUNE	Sample User June
<input type="checkbox"/>	S11U1	S11U1

[Home](#) [Applications](#) [Users](#) [Administration](#)

4.3.1.1.1 Enabling OID Users By default, the users defined for access within mobile server are contained within the mobile server 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 more details, see [Section 4.3.1.6, "Managing OID Users in the Mobile Server"](#).

The mobile server is aware of which users were migrated into OID and marks them as "enabled" for use within Oracle Database Mobile Server. By default, all users created within OID are not "enabled" for use within Oracle Database Mobile Server. All OID users are displayed, but are not enabled for the 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.

4.3.1.1.2 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.

4.3.1.2 Adding New Users

To add a new user, navigate to the Users page and click "Add User". As [Figure 4–2](#) displays, the Add User page appears and lists the requisite criteria to register user properties.

Note: You cannot have a user name with multi-byte characters.

Figure 4–2 Add User Page

Add User

Display Name

User Name

Authentication Mode Internal External

Password


Confirm Password

Privilege

Policy

Register device

Software update Update type

Update date 

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

- [Section 4.3.1.2.1, "Define User Name and Password"](#)
- [Section 4.3.1.2.2, "User Type Assigns Privileges"](#)
- [Section 4.3.1.2.3, "Specify Device Policy for Receiving Updates for this User"](#)

4.3.1.2.1 Define User Name 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.

Table 4–1 (Cont.) Add User Page Description

Field	Description
User Name	<p>Name used to logon to the mobile server. The following are the restrictions when defining the user name:</p> <ul style="list-style-type: none"> ■ Not case sensitive ■ Cannot contain white space characters ■ Maximum length of 28 characters ■ Can contain only alphanumeric characters and special characters '-' (hyphen), '_' (underscore), and '.' (period). ■ Only single-byte characters allowed. You cannot have a user name with multi-byte characters.
Authentication	<p>Select whether this user will be using Oracle Database Mobile Server authentication or if the user will be providing their own.</p> <ul style="list-style-type: none"> ■ Internal—For Oracle Database Mobile Server authentication, select Internal and provide the password used to access the mobile server. ■ External—Select external if this user will be authenticated using External Authentication. See Section 4.3.1.7, "Providing Your Own Authentication for a User" for more information.
Password	<p>For internal authentication, enter password used to logon to the mobile server. When defining, the password must conform to the following restrictions:</p> <ul style="list-style-type: none"> ■ Not case sensitive ■ Cannot contain white space characters ■ Maximum length of 28 characters ■ Must begin with an alphabet ■ Can contain only alphanumeric characters, and special characters of '\$' (dollar sign), '#' (number sign), and '_' (underscore). ■ Cannot be an Oracle database reserved word
Password Confirm	<p>To confirm the above mentioned password for internal authentication, re-enter your password.</p>
Privilege	<p>Lists available privileges for the mobile server user.</p> <ul style="list-style-type: none"> ■ The Administrator privilege allows the user to modify mobile server resources. ■ The Organizer privilege publishes applications. ■ The User privilege enables access for registered users to the mobile server. <p>For a description of each privilege type, see Section 4.1, "What Are the Types of Mobile Server Users?" and Section 4.3.1.2.2, "User Type Assigns Privileges".</p>

4.3.1.2.2 User Type Assigns Privileges Users can be assigned either the administrator or user privileges.

- **Administrator**—The administrator manages the mobile server and its components, publishes and manages applications, and provides application access to groups and users. 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.
- **User**—The User type can access published applications. 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 a mobile 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.

4.3.1.2.3 Specify Device Policy for Receiving Updates for this User Specify the device policy as follows:

Note: For full details on the device policy for receiving updates, see [Section 7.6.1, "Configuring the Device to Receive Required Software Updates"](#)

- **Delete Device:** Normally, when the device associated with the user is de-installed, the device is deregistered in the Mobile Server. If you select "Yes" on this pull-down, then the device object is removed when the device is de-installed. This option is enabled only on a specific user's page, which is shown when you select the user name on the "User's Page" as shown in [Figure 4-1](#).
- **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 pulldown and choose the software update date.

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

4.3.1.3 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 publications.
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 4.3.1.2, "Adding New Users"](#) for more information.
3. Associate the users or groups with the application. See [Section 4.4.1, "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 4.5, "Managing Application Parameter Input (Data Subsetting)" for more information.

4.3.1.4 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 4-1, select the user that you want to duplicate and then click "Create Like". This brings you to a screen where you can enter Display Name, User Name, Authentication and Password, see Table 4-1 for description of these properties. This is the same as when you create a new user, the only difference is that you do not need to specify the Privilege. See Figure 4-3.

Figure 4-3 Create Like

Create Like: JACK

Display Name	<input type="text"/>
User Name	<input type="text"/>
Authentication Mode	<input checked="" type="radio"/> Internal <input type="radio"/> External
Password	<input type="text"/>
Confirm Password	<input type="text"/>

For more information on privileges and device policy, see Section 4.3.1.2, "Adding New Users".

4.3.1.5 Swap Users on a Device

Normally, you permit a single user to use the mobile client on a device. Other users cannot use the device unless one of the following is true:

- All users on the device share the same credentials. This is not secure.
- The mobile client can have any number of users, where each provides their respective credentials. The current user swaps in user's identity for that device by registering the user before using the mobile device. Swapping in a new user de-registers the current user, brings down all of the new user's applications and bootstraps the device with the new user's configuration.

For example, a mobile device that is shared between many employees of a company every day. Each employee selects any device that is pre-loaded with a mobile client installation and uses that device for all daily responsibilities. The employee does not need to retrieve the same device the next day.

Note: All users must be registered with the mobile server before you can swap in a new user.

For Win32, Linux and Windows Mobile platforms, follow below steps to register a new swapped in user:

1. Modify devmgr.ini (mobileclient\bin)
Reset device id:

```
DEVICE_ID=UNKNOWN
```

Update the USER_NAME with a new user (for example, BRIAN)

```
USER_NAME=BRIAN
```

2. Stop currently running dmagent:

```
<mobileclient>\bin\dmagent.exe /c $exit
```

3. Run dmagent to re-register the device with the new user:

```
<mobileclient>\bin\dmagent.exe http://mobile_server_ip/mobile BRIAN <password>
```

Note: If password is not provided, user will be prompted for it.

4.3.1.6 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.

If you decide to use OID users, then—after you install the application server and Oracle Database Mobile Server—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 as described in [Section 4.3.1.6.1, "Migrate Your Users From the Mobile Server Repository to Oracle Internet Directory \(OID\)"](#).
2. Enable OID users for the mobile server. See [Section 4.3.1.1.1, "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".

3. Assign the appropriate application to these users. As with any mobile server user, you must grant access to the appropriate applications. See [Section 4.4.1, "Grant or Revoke Application Access to Users"](#) for more information.

4.3.1.6.1 Migrate Your Users From the Mobile Server Repository to Oracle Internet Directory (OID)

You can use the Oracle Internet Directory (OID), which is the LDAP directory server in the Oracle Identity Management Suite, for storing and retrieving user information instead of the mobile server repository. To use OID, you must migrate all user information from the existing repository into OID.

When you migrate users from a mobile server repository into OID, you cannot have duplicate users in OID. So, if you migrate users from two repositories into a single OID and you have users with the same name, but different passwords on two separate repositories, the user that is first migrated into OID is the one that is valid. The second attempt to migrate an existing user name into OID from a different repository will not migrate and no message is provided. This can be a problem if you have two users in different repositories with different passwords.

Migrate existing users in the repository to OID through the `oiduser` tool, which is located in `MOBILE_HOME\Mobile\Server\bin`. The `oiduser` tool migrates existing users with either randomly-generated passwords or a common password.

Perform the following to migrate your users to ID:

1. Set the `IAS_MODE` parameter in the `mobile.ora` file to `YES`.
2. Migrate the user information using the `oiduser` tool, for either randomly-generated passwords or a common password, as follows:
 - To use randomly-generated passwords for each user, execute the `oiduser` tool without the `-P` option, as follows:

```
oiduser <Mobile Server Repository username> <Mobile
Server Repository password> <OID port number> <OID host name> <OID
password> <OID admin name> <OID subscriber name>
```

For example, the default setting would be:

```
oiduser mobileadmin manager 389 ldap://myhost-pc1.com
welcome1 orcladmin dc=us,dc=oracle,dc=com
```

- To use a common password for all users, provide the common password with the `-P` option, as follows:

```
oiduser <Mobile Server Repository username> <Mobile
Server Repository password> <-P> <common password> <OID port number> <OID
host name> <OID password> <OID admin name> <OID subscriber name>
```

where the common password is specified by you.

All users from the mobile server repository are now migrated to the OID with the required passwords.

If you want to enable Oracle Single Sign On on the mobile server then perform the following:

1. Login to Mobile Manager as the administrator and select the appropriate server.
2. Click on the "Administration" tab.
3. Click "Edit Config File" to edit the `mobile.ora` file for this server.
4. If `SSO_ENABLED` has a hash mark (#) before it, then eliminate the hash mark and set `SSO_ENABLED` to `NO`. Click "Apply".
5. In `mobile.ora`, add the following in section `[EXTERNAL_AUTHENTICATION]`: `CLASS=oracle.lite.resource.OidAuthenticator`
6. Restart both application server and the mobile server.

4.3.1.7 Providing Your Own Authentication for a User

By default, Oracle Database Mobile Server provides authentication through the user name and password to both the mobile server. However, if you want to add your own external authentication for the user, such as a fingerprint pad and so on, then you can use APIs to designate what authenticator to use.

For logging on and access to the mobile server, external authentication can be added. For full details, see Section 8.1, "Providing Your Own Authentication Mechanism for Authenticating Users for the Mobile Server" in the *Oracle Database Mobile Server Developer's Guide*.

4.3.2 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

The screenshot shows the "Add Group" page. It features a "Group Name" input field. Below it is a "Policy" section with three settings: "Register device" is a dropdown menu set to "TRUE"; "Software update" has two radio buttons, with "Update type" selected and set to "Minor update" (a dropdown menu), and "Update date" set to "8/15/11" with a calendar icon; and "Cancel" and "OK" buttons at the bottom right.

Enter the new group name in the "Group Name" field. The device policy for the group has the same options as for a single user. For more information on device policy, see [Section 4.3.1.2.3, "Specify Device Policy for Receiving Updates for this User"](#). When finished, click **OK**.

4.3.3 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.

4.4 Managing Access Privileges for Users and Groups

The mobile server administrator grants 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 4.4.1, "Grant or Revoke Application Access to Users"](#)
- [Section 4.4.2, "Include or Exclude Users from Group Based Access"](#)
- [Section 4.4.3, "Grant or Revoke Application Access to Groups"](#)

4.4.1 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 4.4.1.1, "Grant Application Access to Users"](#)
- [Section 4.4.1.2, "Revoke Application Access to Users"](#)

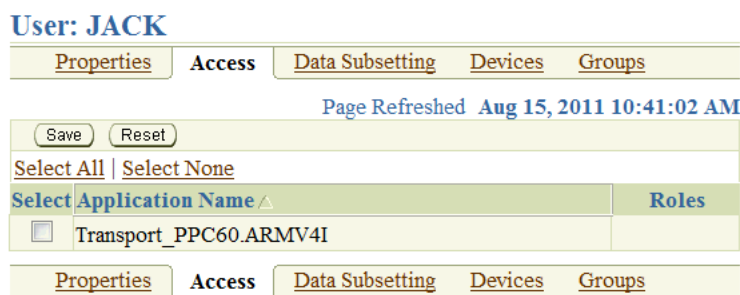
4.4.1.1 Grant Application Access to Users

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 Transport_PPC60.ARMV4I application.

Figure 4–5 Granting Application Access



4.4.1.2 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 4.4.2, "Include or Exclude Users from Group Based Access"](#).

4.4.2 Include or Exclude Users from Group Based Access

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

- [Section 4.4.2.1, "Include Users in a Group"](#)
- [Section 4.4.2.2, "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.

4.4.2.1 Include Users in a Group

To include users into a group, do the following:

1. Navigate to the "Users" page. Click the name 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 4.4.1, "Grant or Revoke Application Access to Users"](#). However, instead of selecting a particular user, select the group instead.

4.4.2.2 Exclude Users from a Group

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

1. Navigate to the "Users" page. Click on the name 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

Group: PUBLIC GROUP

[Properties](#) [Users](#)

Page Refreshed Aug 15, 2011 10:42:58 AM

[Save](#) [Reset](#)

[Select All](#) | [Select None](#)

Select	User Name ▲	Display Name
<input type="checkbox"/>	AA	aa
<input type="checkbox"/>	JACK	Sample User Jack
<input type="checkbox"/>	JANE	Sample User Jane
<input type="checkbox"/>	JOHN	Sample User John
<input type="checkbox"/>	JUNE	Sample User June
<input type="checkbox"/>	S11U1	S11U1

[Properties](#) [Users](#)

4.4.3 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, "Granting Application Access to Users and Groups"](#) for directions.

4.5 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 4.4, "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".

4.6 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 a user the mobile client software on an installation CD, and the user installs the mobile client software offline, then the installation does not register the device automatically—since it is not connected to mobile server. See [Chapter 8, "Offline Instantiation for Mobile Clients,"](#) for more information. For each user that install the mobile client software offline, 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.

- Click "Add". The "Create Device" screen (as shown in [Figure 4-7](#)) appears.

Figure 4-7 Manually Add Device to User

Create Device

Language

Name

Device Name

Platform

Address

Device Address

Network Provider

- Enter the device information, as described in [Figure 4-2](#), and click "OK" to add the device for this user:

Table 4-2 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.
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. 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 Configuring How the Device Receives Software Updates for the User

You can control whether a new version of an application software is downloaded on each client. See [Section 4.3.1.2.3, "Specify Device Policy for Receiving Updates for this User"](#) for full details on how the device policy is implemented for receiving updates for this user.

Managing Synchronization

The mobile server administrator uses the Data Synchronization Manager to manage synchronization tasks. This chapter includes the following:

- [Section 5.1, "How Does the Synchronization Process Work?"](#)
- [Section 5.2, "Initiate Synchronization of the Mobile Client"](#)
- [Section 5.3, "User Scenarios for Synchronization"](#)
- [Section 5.4, "Managing the Sync Server from the Data Synchronization Home Page"](#)
- [Section 5.5, "Using Automatic Synchronization"](#)
- [Section 5.6, "Configuring Data Synchronization For Farm or Single Mobile Server"](#)
- [Section 5.7, "Resuming an Interrupted Synchronization"](#)
- [Section 5.8, "Register a Remote Oracle Database for Application Data"](#)
- [Section 5.9, "Improving Performance for Multiple Clients that Use the Same Read-Only Data With a Cached User"](#)
- [Section 5.10, "Synchronizing to a File With File-Based Sync"](#)
- [Section 5.11, "Managing Trace Settings and Trace Files"](#)
- [Section 5.12, "Browsing the Repository for Synchronization Details"](#)
- [Section 5.13, "Monitoring and Analyzing Performance"](#)

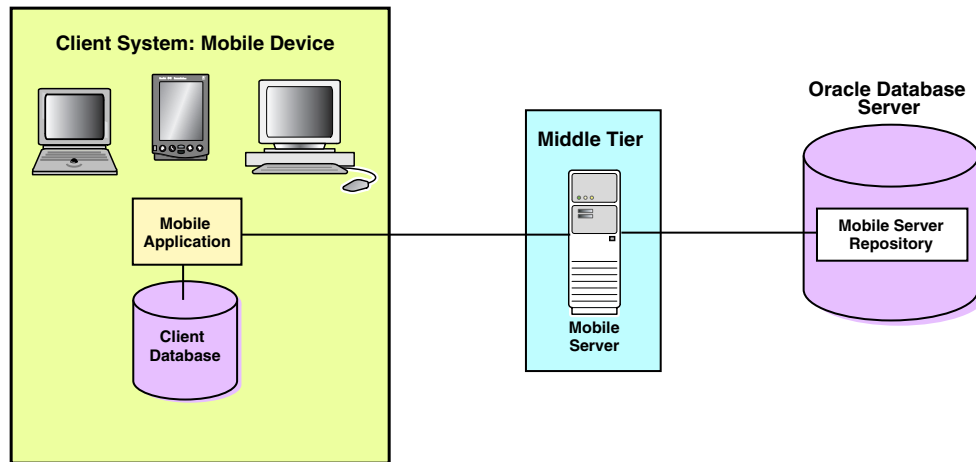
5.1 How Does the Synchronization Process Work?

With Oracle Database Mobile Server, you can create an application where multiple users enter data on their client devices and the data is synchronized with a back-end Oracle database. In addition, only the data designated for each user is downloaded from the server to the client's device.

The mobile server uses synchronization to replicate data between the mobile clients with their client databases and the application tables, which are stored on a back-end Oracle database. The Oracle Database Mobile Server architecture consists of the mobile client, the mobile server as the middle tier, and the mobile server repository in the back-end Oracle database server. The mobile client contains the client database and client applications, which resides on the mobile device. The mobile server acts as the middle tier to coordinate the synchronization process and provide management tools for the administrator. The mobile server repository resides in the back-end Oracle database server and is where all data from all users is stored.

Figure 5–1 shows a simplified architecture of the Oracle Database Mobile Server synchronization tiers:

Figure 5–1 General Synchronization Architecture



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 Mobile Server, which is used for multiple clients. In order to accommodate 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.

In reality, our synchronization process uses asynchronous communication and queues to ensure that multiple users can be synchronizing at the same time as well as ensuring data integrity and performance. By using queues and asynchronous communication, the user can queue the updates for processing and then download any updates from the server without ever locking the database. The only downside is that the changes are not immediately updated. Instead, the updates occur when the In Queue is read and processed on the server-side.

The following graphic and steps shows the full details and the additional components that work in conjunction to enable a seamless synchronization from multiple clients. This graphic introduces the following new components:

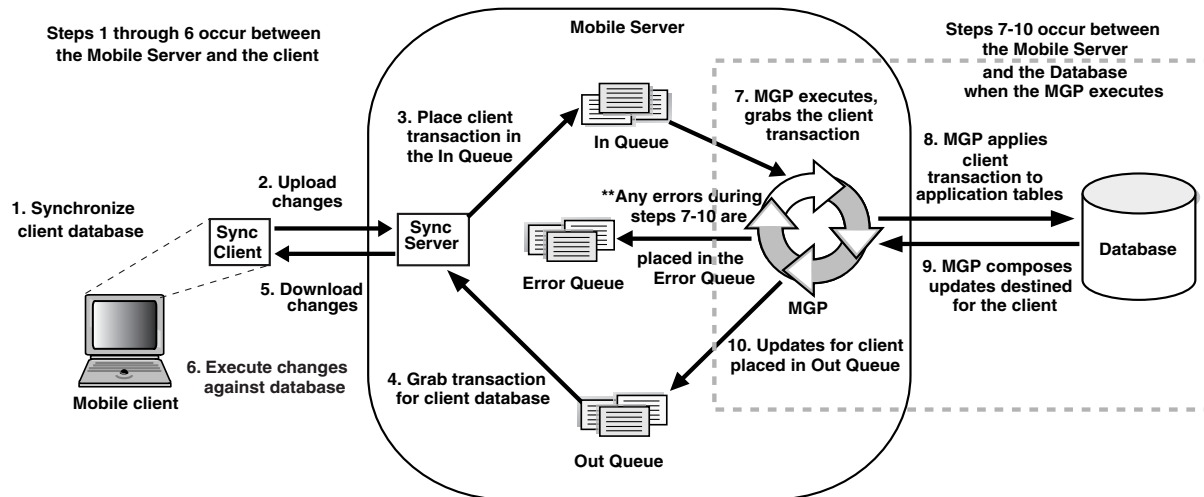
- **Sync Client and Sync Server:** The Sync Client and Sync Server work in conjunction to upload mobile client changes to the mobile server In Queue. The Sync Client resides on the mobile client; the Sync Server resides on the mobile server. The Sync Server places the client modifications into the In Queue and downloads any new data from the server for the client from the Out Queue.
- **Message Generator and Processor (MGP):** A background process called the Message Generator and Processor (MGP) runs in the same tier as the mobile server, periodically collects all the uploaded changes from the multiple mobile users out of the In-Queue and applies these changes to the repository in the server database. The MGP executes independently and periodically based upon an interval specified in the Job Scheduler in the mobile server.

The MGP prepares any modifications that are sent to each mobile user and places them into the Out Queue. The next time the mobile user synchronizes with the

mobile server, these changes can be downloaded to the client and applied to the client database.

Figure 5–2 describes how each of these components interact to complete the asynchronous, queue-based synchronization for each client:

Figure 5–2 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 PDA, a Windows platform client or a supported Linux platform client.
2. The 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 scalable 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 database.
6. The mobile client applies all changes for client database.
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.

Synchronization involves two components: the Sync Client/Server and the MGP process, which 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, set the `DO_APPLY_BFR_COMPOSE` parameter to NO. 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

MOBILE.ORA Parameter	Description
<code>DO_APPLY_BFR_COMPOSE</code>	Setting <code>DO_APPLY_BFR_COMPOSE</code> to no, the Compose executes. However, by default, this parameter is set to yes, so that if data is in the in queue, MGP will execute a second Apply to commit all user data and then will execute the Compose.
<code>SKIP_INQ_CHK_BFR_COMPOSE</code>	Setting <code>SKIP_INQ_CHK_BFR_COMPOSE</code> 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 2.1.4 "MGP Compose Postponed Due to Transaction in the In-Queue"](#) in the *Oracle Database Mobile Server Troubleshooting and Tuning Guide*.

5.2 Initiate Synchronization of the Mobile Client

You can initiate synchronization of the client using one of the following methods:

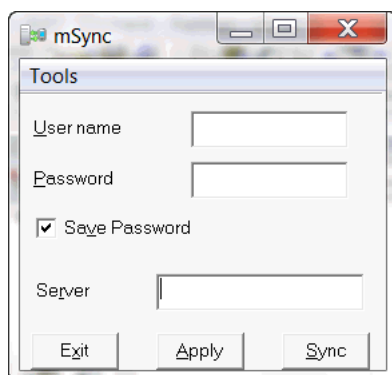
- You can initiate synchronization of the client using the mSync GUI.
- Implement synchronization within your application using the synchronization APIs (see Chapter 2, "Synchronization" and Chapter 3, "Managing Synchronization on the Mobile Client" in the *Oracle Database Mobile Server Developer's Guide* for more information)

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

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

The mSync GUI is shown in [Figure 5-3](#).

Figure 5-3 Using the mSync GUI to Initiate Synchronization



To bring up the mSync GUI, execute `msync.exe` on Windows Mobile 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:

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

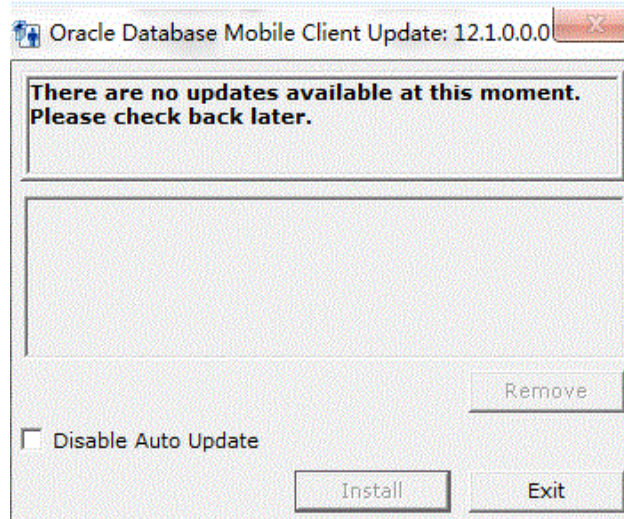
Note: See [Section 4.3.1.2.1, "Define User Name and Password"](#) for conventions for creating the user name or password.

- Check the "Save Password" box if you want the password saved for future requests.
- Mobile Server URL, which is `http://<mobile_server_hostname>:<port_number>`, replace 'http' with 'https' in case SSL is used. If SSL is not used, 'http://' can be skipped.

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

If there are software updates that are waiting to be downloaded to the client, the update tool is automatically executed after the end of the synchronization process, see [Figure 5-5](#).

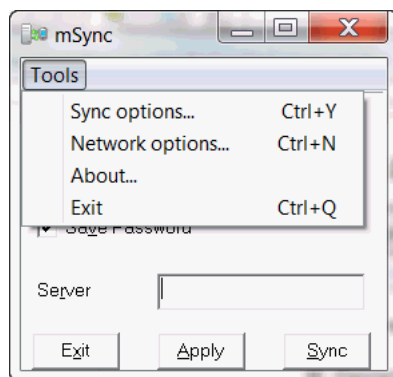
Figure 5-4 Update Tool



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

You can also modify the tool options by selecting the "Tools Selection" at the bottom of the UI, as shown in [Figure 5-5](#).

Figure 5-5 The mSync Tools Selection



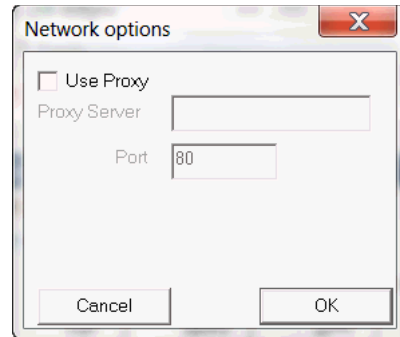
The following sections describe the Tools options:

- [Section 5.2.1, "Network Options for MSync Tool"](#)
- [Section 5.2.2, "Sync Options for MSync Tool"](#)
- [Section 5.2.3, "Use Mobile Client Tools on Linux"](#)

5.2.1 Network Options for MSync Tool

Figure 5–6 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 5–6 The mSync Network Options Selection



5.2.2 Sync Options for MSync Tool

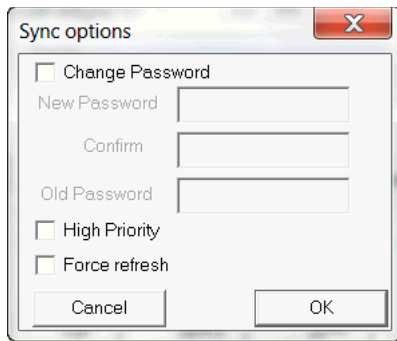
Figure 5–7 displays the "Sync Options" screen where you can specify the following:

- 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.
- High Priority—Select this checkbox to specify synchronizing only High Priority data. This specifies under what conditions the different priority records are synchronized. By default, the value is LOW, which is synchronized last. If you have a very low network bandwidth and a high ping delay, you may only want to synchronize your HIGH priority data.

When you select this checkbox, you are enabling pre-defined high priority records to be synchronized first. This only for those publication items that have specified a restricting predicate. See Section 1.2.10, "Priority-Based Replication" in the *Oracle Database Mobile Server Troubleshooting and Tuning Guide* for more information.

- Force Refresh—The force refresh option is an emergency only synchronization option. Check this option when a client is corrupt or malfunctioning, so that you decide to replace the mobile client data with a fresh copy of data from the enterprise data store with the forced refresh. When this option is selected, any data transactions that have been made on the client are lost.

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

Figure 5–7 The mSync Options Selection

5.2.3 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.

5.3 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 5.5, "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 5.2, "Initiate Synchronization of the Mobile Client"](#)
 - If the application is going to initiate the synchronization, use the synchronization APIs, as described in Chapter 3, "Managing Synchronization on the Mobile Client" in the *Oracle Database Mobile Server 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 5.9, "Improving Performance for Multiple Clients that Use the Same Read-Only Data With a Cached User"](#) for more details.
- You may save all synchronization transactions in an encrypted file. There are times when you do not have network access to the mobile server, but there is a way you can manually carry a file between the mobile server and the client. In this instance, you may want to use File-Based Sync, which saves all transactions in an encrypted file either for the upload from the client for the mobile server or the download from the mobile server for the client. See [Section 5.10, "Synchronizing to a File With File-Based Sync"](#) for more details.
- You may decide that it would be more performant to store your application data on an Oracle database that is separate from the Oracle database that contains the mobile server repository. In this case, you can have multiple back-end databases, where the mobile server repository exists on the main database and the data for one or more applications may exist on the main database or another database of your choosing. For more information, see [Section 5.8, "Register a Remote Oracle Database for Application Data"](#).

5.4 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 5-2](#), during every synchronization session, the Sync Server uploads client transactions from the client database and places them within the In-Queues. The Sync Server then downloads any server-side transactions from the Out-Queues to the client database.

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

- [Section 5.4.1, "Starting/Stopping the Sync Server"](#)
- [Section 5.4.2, "Checking Synchronization Alerts"](#)
- [Section 5.4.3, "Managing Sync Sessions"](#)
- [Section 5.4.4, "Displaying Operating System \(OS\) and Java Virtual Machine \(JVM\) Information"](#)

5.4.1 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 5-8](#).

Figure 5–8 Data Synchronization Home Page

Page Refreshed Aug 16, 2011 9:24:38 AM

Home Performance Administration Repository MGP Platforms

General



Status **Up**
 Status Days **0.9**
 Status Date **Aug 15, 2011 11:34:50 AM**
 History Sessions **16**
 Host **PC5**
 Related Links [Job Scheduler](#)

Alerts

Select	Name	Severity	Alert Triggered
	(No items found)		

Active Sessions

Search

Select	ID	Device User	Type	Phase	Start Time	Duration (seconds)	Upload Duration (seconds)	Upload Record Count	Download Duration (seconds)	Download Record Count	Complete Refresh PubItem Count
	(No items found)										

Home Performance Administration Repository MGP Platforms

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.

5.4.2 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

Table 5–2 (Cont.) Alert Types

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

5.4.3 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 5–9 shows the "Synchronization History Sessions" page.

Figure 5–9 Synchronization History Sessions Page

Page Refreshed Aug 16, 2011 9:32:40 AM

Synchronization History Sessions

Search

User:

Device Type: All

Server Result: All

Device Result: All

From: Date: 8/9/11
Example: 10/31/03
 Time: 9:30 AM

To: Date: 8/23/11
Example: 10/31/03
 Time: 9:30 AM

Time Zone: Pacific Standard Time

Results

Select	ID	User	Device Type	Server Result	Device Result	Synchronization Finish Time	Duration (seconds)	Upload Duration (seconds)	Upload Record Count	Download Duration (seconds)	Download Record Count	Complete Refresh PubItem Count
<input checked="" type="radio"/>	167	JOHN	SQLite WIN32	✓		AUG 15, 2011 06:29:03 PM	1	1	0	0	0	0
<input type="radio"/>	164	JOHN	SQLite WIN32	✓		AUG 15, 2011 06:29:00 PM	0	0	0	0	0	0
<input type="radio"/>	161	AA	SQLite WIN32	✓		AUG 15, 2011 11:00:59 AM	1	1	0	0	0	0
<input type="radio"/>	140	S11U1	SQLiteJava	✓		AUG 10, 2011 04:31:49 PM	0	0	0	0	0	0
<input type="radio"/>	137	S11U1	SQLiteJava	✓		AUG 10, 2011 04:28:23 PM	0	0	0	0	0	0

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.

5.4.4 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 5–10, 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 5–10 Host Page

General		JVM	
Name	PCS	Java Version	1.6.0_23
IP	130.35.26.20	Total Heap Memory (MB)	97
Hardware	x86	Free Heap Memory (MB)	64
Operating System	Windows 7 6.1		
OS User Name	bber		
Environment Variables			
Java Class Path	C:/glassfish311/glassfish/modules/glassfish.jar;C:/glassfish311/glassfish/lib/monitor/flashlight-agent.jar		

5.5 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 Mobile Server 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 item level, see [Section 5.5.2, "Start, Stop, or Get Status for Automatic Synchronization"](#) or [Section 2.2, "Enabling Automatic Synchronization"](#) in the *Oracle Database Mobile Server Developer's Guide*.

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

Note: Within a publication, you can have one or more publication items. If automatic synchronization is enabled for one of the publication items, by default all the publication items in the same publication would be enabled with automatic synchronization.

For more information on how to specify rules for all enabled publication items, see Section 2.2, "Enabling Automatic Synchronization" in the *Oracle Database Mobile Server 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 5.5.1, "Specifying Platform Rules for Automatic Synchronization"](#) for more information.

Automatic synchronization is based on a different model than manual synchronization. Automatic synchronization operates on a transactional basis. Thus, when the conditions are correct, any new data transactions are uploaded to the server, in the order of the specified priority for the data. In the manual synchronization model, you can synchronize all data or use the selective sync option, where you can detail only certain portions of the data to be synchronized. The selective sync option is not supported in automatic synchronization, since we are no longer concerned with synchronization of only a subset of data.

The following provides more information about your automatic synchronization:

- [Section 5.5.1, "Specifying Platform Rules for Automatic Synchronization"](#)
- [Section 5.5.2, "Start, Stop, or Get Status for Automatic Synchronization"](#)
- [Section 5.5.3, "How the Automatic Synchronization Transaction is Retried"](#)

5.5.1 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 Platforms 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:

- On the Data Synchronization page, select "Platforms". [Figure 5–11](#) shows the settings for automatic synchronization on each platform.

Figure 5–11 Platforms for Automatic Synchronization

Data Synchronization

Page Refreshed Aug 16, 2011 9:40:51 AM			
Home	Performance	Administration	Repository
MGP	Platforms		
Platform Name	Events	Conditions	Networks
WINCE	Not Defined	Not Defined	Not Defined
WINDOWS	Not Defined	Not Defined	Not Defined
Linux x86	Not Defined	Not Defined	Not Defined
Java	Not Defined	Not Defined	Not Defined

- Select the platform name to modify the automatic synchronization platform settings. Figure 5–12 shows the screen for the platform-based rules. There are three tabs: "Events", "Conditions", and "Networks".

Figure 5–12 Platform-Based Events for Automatic Synchronization

Platform: WINDOWS

Events Conditions Networks

Automatic synchronization events

Synchronize when the network bandwidth is greater than bit/s

Synchronize when the battery level drops to %

Synchronize when AC power is detected

Synchronize at a specified time or time interval

Specify time interval

Hour Minutes

Specify Time

Select Hour	Minutes

Events Conditions Networks

- Configure the Events. Figure 5–12 shows the "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 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.

However, you must enter the time in Coordinated Universal Time (UTC) for a specific time as it gives us a common base. Every timezone can be expressed as an offset from UTC (that is, UTC+4) and by doing this the

server does not need to know about each client's respective timezones. For example, if you want something to happen at 8:00 A.M your time, you must specify that time in UTC and then enter that into the server.

- 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 5–13](#) displays the "Conditions" screen.

If an Automatic Synchronization is about to start, Oracle Database Mobile Server 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 5–13 Platform-Based Conditions for Automatic Synchronization

Platform: WINDOWS

Page Refreshed Aug 16, 2011 9:43:53 AM

Events **Conditions** Networks

System Conditions

Synchronize only if the battery level is greater than %

Revert Apply

Data/Network Conditions

Edit Delete

Select	Data Priority	Minimum Network Bandwidth (bits/sec)	Maximum Ping Delay (ms)	Include Dial-up Networks?
<input checked="" type="radio"/>	High			
<input type="radio"/>	Low			

Events **Conditions** Networks

The following conditions must be true for this platform 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 5-14](#) displays the fields that you can specify for this condition.

Figure 5-14 Editing the Data Priority Condition

Edit Condition

Data Priority High

Minimum Network Bandwidth (bits/sec)

Maximum Ping Delay (ms)

Include Dial-up Networks? No ▾

Cancel Apply

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 5–15 Add Network Information for Automatic Synchronization**Platform: WINDOWS**

Page Refreshed Aug 16, 2011 9:46:10 AM

Events Conditions **Networks**

Always-on Networks

Proxy Server

Port

Revert Apply

Dial-up Networks

Add Dial-up Network

Select			
<input type="checkbox"/>			

Events Conditions **Networks**

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 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 5–16 Add Dial-Up Network Information**Add Dial-up Network**

Network Name

Proxy Server

Port

Cancel Reset Apply

If you are required to provide proxy configuration for any dial-up network, then configure the following so that Oracle Database Mobile Server 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.

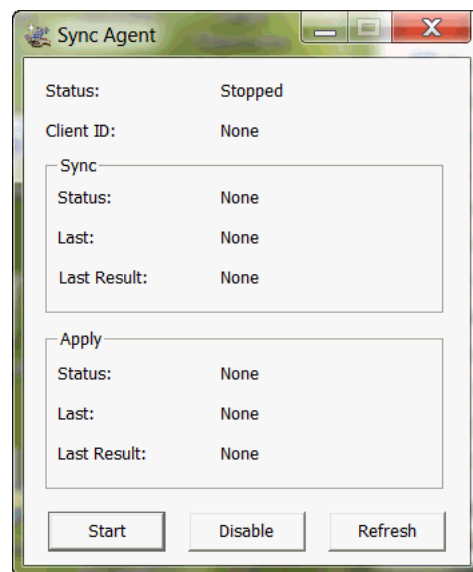
- Port—Provide the port number of the proxy server.

5.5.2 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 `autosync` executable in a command prompt. Figure 5–17 shows the Sync Agent UI.

Note: You can also start and stop automatic synchronization using programmatic APIs. For full details, see Section 3.2, "Managing Automatic Synchronization on the Mobile Client" in the *Oracle Database Mobile Server Developer's Guide*.

Figure 5–17 Managing Automatic Sync Agent



The Sync 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 Sync Agent is "Disabled". Enable the Sync Agent by clicking "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". To see the status of any existing or the last automatic synchronization, click "Refresh".

5.5.3 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.

5.6 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 server 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 `mobile.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 Files for the Mobile Server"](#) for a description of each of these parameters.

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

5.7 Resuming an Interrupted Synchronization

With client/server networking, communication may be interrupted by unreliable network conditions, physical disconnections, limited transport bandwidth, and so on. To efficiently cope with these conditions, the transport protocol between the client and server resumes a synchronization session from the last acknowledged byte. For example, the client starts to upload 10 MB of data and the connection fails after sending 9 MB of the data. In this instance, the client does not resend the 9 MB that was

acknowledged, but resumes the synchronization by uploading the last 1 MB of data. The resume feature works the same for both the upload and download phases of the transport.

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

Configure the resume feature parameters, as follows:

- [Section 5.7.1, "Defining Temporary Storage Location for Client Data"](#)
- [Section 5.7.2, "Controlling Server Load"](#)

5.7.1 Defining Temporary Storage Location for Client Data

By default, the client data is buffered in memory and maximum of 16 MB is allocated for the buffering. If more space is needed, new clients are blocked until space is freed. Alternatively, you can configure where the client data is temporarily stored and how much space to allocate with the `RESUME_FILE` and `RESUME_FILE_SIZE` parameters in the `CONSOLIDATOR` section of the `mobile.ora` file on the mobile server, as follows:

```
RESUME_FILE=d:\path\file
RESUME_FILE_SIZE =NNN (MB)
```

Setting the `RESUME_FILE_SIZE` parameter configures the amount of memory allocated for the buffering. Setting `RESUME_FILE` allows using a disk file instead of RAM, which is more efficient if JDK 1.4 or later is installed and memory mapping can be used.

If there are multiple disks available on the mobile server host, one spool file should be created per disk to optimize performance. You can specify several spool files with multiple `RESUME_FILE` and `RESUME_FILE_SIZE` parameters, each designated with a unique suffix, as follow: `RESUME_FILE_2`, `RESUME_FILE_SIZE_2`. The maximum number of files is determined by the operating system and hardware for the mobile server. The maximum size for each file is 2047 MB.

Normally, 64 KB blocks are used to buffer client data. Resume block size can be specified in KB, with the `RESUME_BLOCKSIZE` parameter. If you are using disk files to minimize fragmentation, then the block size should be specified as a larger number.

5.7.2 Controlling Server Load

If too many clients connect to a mobile server at once, the mobile server can become overloaded, run out of memory, or have poor performance when responding to the clients.

The `RESUME_MAXACTIVE` parameter controls the maximum number of connections that 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.

After `RESUME_MAXACTIVE` has reached, configure `RESUME_MAX_WAIT` for the number of minutes a new client should wait before returning with error message. This parameter is configured in minutes.

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.

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 discard all data before that offset. The fault value is 1024 KB.

These parameters are all configured within the `mobile.ora` file on the mobile server.

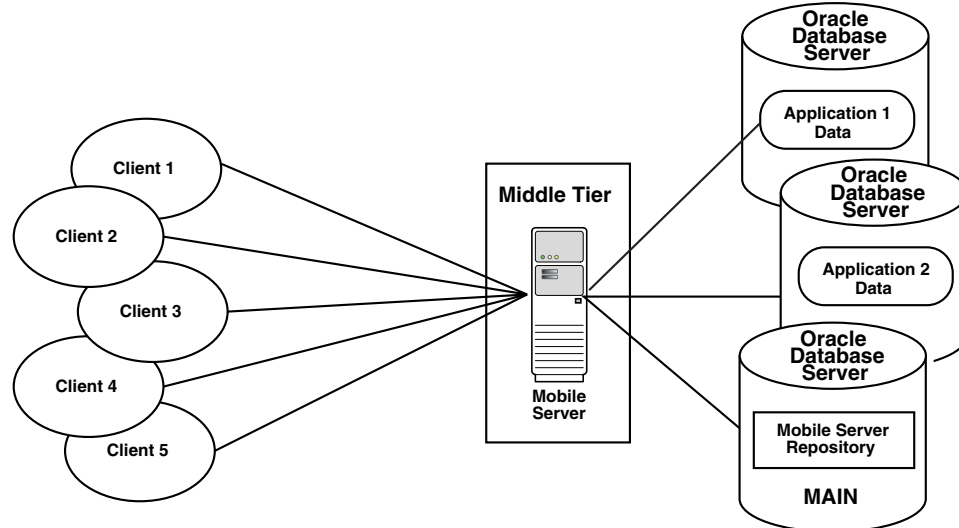
5.8 Register a Remote Oracle Database for Application Data

By default, the mobile server repository metadata and the application schemas are present in the same database. However, it is possible to place the application schemas in a database other than the Main database where the mobile server repository exists. This can be an advantage from a performance or administrative viewpoint.

Thus, you can spread your application data across multiple databases.

Note: We refer to the database where the application schema resides as remote because it is separate from the Main database that contains the mobile server repository. It does not mean that the database is geographically remote. It can be local or remote. For performance reasons, the mobile server must have connectivity to all databases involved in the synchronization—Main and remote.

Figure 5–18 Separating Application Data from Mobile Server Repository



You can register one or more remote Oracle databases that host the application data. Once registered, you can specify application schemas on these databases during publication creation. Synchronization is executed on a per publication basis rotating through the databases.

To use an Oracle database other than the Oracle database used for the mobile server repository, you must perform the following:

1. Register the remote Oracle database as described in [Section 5.8.1, "Register or Deregister a Remote Oracle Database for Application Data"](#).

2. When creating the publication item, specify the URL of the Oracle database for storing the application data. All data for a single application must be contained in the same Oracle database. In MDW, when you select New-Project, you can specify the URL for the remote database for the publication. For more information, see Section 4.2, "Create a Project" in the *Oracle Database Mobile Server Developer's Guide*.

In order for the synchronization to complete successfully, both the application database and the mobile server repository database must be up.

Note: For details on how to use Consolidator APIs or to modify callbacks to use a remote Oracle database, see Section 2.15, "Register a Remote Oracle Database for Application Data" in the *Oracle Database Mobile Server Developer's Guide*.

5.8.1 Register or Deregister a Remote Oracle Database for Application Data

If you want your application data to be in a database other than where the mobile server resides, then you can register a remote database through the Mobile Manager. However, once registered, the remote database cannot be used by any other installation of Oracle Database Mobile Server.

Publications and publication items can be defined against application data in the remote database. The Mobile Manager manages the Main database and all registered application databases in the same way.

Note: All registered databases must be up and running for proper operation. If a database is already used by a separate Oracle Database Mobile Server installation, either as a mobile server repository or as an application repository, then the registration fails. An application repository can be deregistered from one installation and then registered to a different installation.

To register the Oracle database, navigate to Data Synchronization->Repository. The bottom section lists the registered Oracle databases, where MAIN is the Oracle database that contains the mobile server repository.

1. As shown in [Figure 5-19](#), click the "Register Database" button to register a new database.

Figure 5–19 Register or Deregister Oracle Database for Application Data

Page Refreshed Aug 16, 2011 9:50:40 AM

Home	Performance	Administration	Repository	MGP	Platforms
----------------------	-----------------------------	--------------------------------	----------------------------	---------------------	---------------------------

<p>Users and Publications</p> <p>Users (6) Publications (8) Publication Items (14)</p>	<p>Queues</p> <p>In Queue (0) Out Queue (0) Error Queue (0)</p>
---	--

Databases

Deregister	Register Database		
Show All Details Hide All Details			
Select Details	Name	Description	Instance
<input type="radio"/>	Show MAIN	Main Repository for Mobile Server	orcl

Home	Performance	Administration	Repository	MGP	Platforms
----------------------	-----------------------------	--------------------------------	----------------------------	---------------------	---------------------------

2. Enter the Oracle database information as follows:

- **Name**—An identifying name for the database where the application schema resides. Once defined, this name cannot be modified. This name must be unique across all registered database names.

Note: A log is generated for each remote database application in the `ORACLE_HOME/Mobile/Server/<DB_NAME>/apprepository.log` file, where the `<DB_Name>` is this identifying name.

- **Description**—A user-defined description to help identify this database.
- **JDBC URL**—The JDBC URL can be one of the following formats:
 - * The URL for a single Oracle database has the following structure:
`jdbc:oracle:thin:@<host>:<port>:<SID>` for Oracle Database 10g and Oracle Database 11g, `jdbc:oracle:thin:@//<host>:<port>/<ServiceName>` for Oracle Database 12c.

* The JDBC URL for an Oracle RAC database can have more than one address in it for multiple Oracle databases in the cluster and follows this URL structure:

```
jdbc:oracle:thin:@(DESCRIPTION=
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=TCP) (HOST=PRIMARY_NODE_HOSTNAME) (PORT=1521))
    (ADDRESS=(PROTOCOL=TCP) (HOST=SECONDARY_NODE_HOSTNAME) (PORT=1521))
  )
  (CONNECT_DATA=(SERVICE_NAME=DATABASE_SERVICENAME)))
```

- **Schema Name**—The application schema name.
- **Password**—The administrator password is used to logon to the database. The administrator name is the same as what was defined for the main database.

When defining, the password must conform to the following restrictions:

- not case sensitive

- cannot contain white space characters
 - maximum length of 28 characters
 - must begin with an alphabet
 - can contain only alphanumeric characters, and special characters of '\$' (dollar sign), '#' (number sign), and '_' (underscore)
 - cannot be an Oracle database reserved word
3. Click "Apply". The Oracle database is now registered with the mobile server and can be specified in a publication item.

After you register a database, Oracle Database Mobile Server creates the application database repository in the application database. The application database repository includes the schema and related consolidator tables. This is similar to the mobile server repository creation on the MAIN database.

When registering an application database, the user must provide the correct password for the repository schema. The password used is the user password. However, if the password provided is incorrect, then the administrator user name and password are requested.

Edit Oracle Database Details

To edit, select the name of the desired registered Oracle database in the list.

You can edit the description and password for any registered database. The password is used to logon to the application database. Before you can modify the password here, you must first modify the password on the database itself. Once modified on the database, then modify it in the mobile server as well on this screen.

DeRegister Oracle Database

Before you can deregister any database, you must first drop all associated publication and publication items. To deregister, select the radio button next to the desired database and click "Deregister". Click "Yes".

5.9 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 is 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 one user for each group of clients that subscribe to the same subset of data. 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 8, "Offline Instantiation for Mobile Clients"](#) 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.

For cached user, if you try to synchronize for multiple clients at the same time, the server will synchronize serially and will block all clients while a single client is updated.

If you are using the cached user feature, you cannot enable automatic synchronization. If you do enable automatic synchronization, only one client will be notified to initiate an automatic synchronization.

5.10 Synchronizing to a File With File-Based Sync

There are times when you do not have network access to the mobile server, but there is a way you can manually carry a file between the mobile server and the client. In this instance, you may want to use File-Based Sync, which saves all transactions in an encrypted file either for the upload from the client to the mobile server or the download from the mobile server to the client. File-Based Sync cannot be automatic, because files need to be manually transferred from client to server and back.

Once saved within the encrypted file, the file is manually transported and copied onto the desired recipient—whether mobile client or mobile server. This file is uploaded and the normal synchronization steps are performed.

Most of the activity on the mobile client and the mobile server is the same as with a network synchronization. The only difference is that instead of the data being transmitted over the network, it is placed into a file that is transported in another manner.

The following sections describe the activity necessary on the mobile client and the mobile server for file-based synchronization:

- [Section 5.10.1, "Upload Synchronization Transactions to an Encrypted File on the Mobile Client"](#)

- [Section 5.10.2, "Apply and Compose Mobile Client Transactions on the Mobile Server"](#)
- [Section 5.10.3, "Download Composed Transactions from Mobile Server to the Mobile Client"](#)
- [Section 5.10.4, "Troubleshooting File-Based Synchronization Scenarios"](#)

5.10.1 Upload Synchronization Transactions to an Encrypted File on the Mobile Client

Perform the following on the mobile client to upload all synchronization transactions to an encrypted file destined for the mobile server:

1. Enable file-based synchronization with one of the following methods:
 - When using the mSync UI tool, select the File-Based Sync option off the Tools menu. Perform the steps described in Section 3.3.3, "Sync to a File Using File-Based Sync" in the *Oracle Database Mobile Server Mobile Client Guide*, setting the radio button to Send.
 - Enable file-based synchronization programmatically with the set Synchronization Property APIs. For upload, set the filename and the sync direction to Send. See the appropriate programming language section in Chapter 3, "Managing Synchronization on the Mobile Client" in the *Oracle Database Mobile Server Developer's Guide*.
2. For manual synchronization, initiate the synchronization.

If the synchronization fails during writing of the synchronization file, then the file may be deleted and the synchronization can be restarted. When restarted, the synchronization engine recreates the same synchronization file. If the failure occurs again, then it is most likely a hardware issue that must be resolved, such as not enough storage, database corruption or other hardware errors.

Note: When the mobile server uploads the synchronization file or receives an upload from a network synchronization, it downloads any new data for the client as well as sends an acknowledgement for the transactions it received.

If, during a synchronization request, the client has not received or downloaded any acknowledgement from the mobile server from the last upload, then the client will send all unacknowledged, previously uploaded transactions as well as any new transactions.

3. Once the file is created in the directory specified, transport it to the mobile server using removable media or any other appropriate manner.

5.10.2 Apply and Compose Mobile Client Transactions on the Mobile Server

To perform the apply and compose function on the mobile server for mobile client transactions that are saved in an encrypted synchronization file, perform the following on the mobile server:

Note: Only the user is allowed to upload the synchronization file from the client with the user's id. All other users will be denied permission for the upload.

1. On the "Mobile Manager Home" page, before you log in your mobile server, select the "File-Based Sync" tab.
2. Enter the user name and password of the client user. Click "Logon".
3. Using the "Browse" button, locate and identify the encrypted synchronization file on the mobile server machine. Click "Sync". The file uploads to the mobile server.
The mobile server starts the MGP to perform the apply/compose phase for the mobile client based upon the uploaded synchronization file. If this fails after several attempts, then the file may be corrupted. You can re-create the file and try again.
4. A screen prompts the user for the name and directory for the synchronization file for the mobile client. This file downloads all composed transactions meant for the user to the designated file from the mobile server. Enter the name and directory for the file. Use your removable media to transport this file to the mobile client.

5.10.3 Download Composed Transactions from Mobile Server to the Mobile Client

To download all transactions from the mobile server to the mobile client, perform the following on the mobile client:

1. Enable file-based synchronization with one of the following methods:
 - When using the mSync UI tool, select the File Based Sync option off the Tools menu. Perform the steps described in Section 3.3.3, "Sync to a File Using File-Based Sync" in the *Oracle Database Mobile Server Mobile Client Guide*, setting the radio button to "Receive". Browse for the encrypted synchronization file that was transported from the mobile server.
 - Enable file-based synchronization programmatically with the set Synchronization Property APIs. For download, browse for the file that was transferred from the mobile server and set the sync direction to "Receive". See the appropriate programming language section in Chapter 3, "Managing Synchronization on the Mobile Client" in the *Oracle Database Mobile Server Developer's Guide*.
2. For manual synchronization, initiate the synchronization.

If an error occurs while receiving the synchronization file, you may restart the processing again. The client verifies that the transaction has not been processed.

5.10.4 Troubleshooting File-Based Synchronization Scenarios

When you have a disconnected synchronization, you may run into one of the following scenarios:

- [Section 5.10.4.1, "Normal Network Synchronization Occurs During File Upload"](#)
- [Section 5.10.4.2, "Conflict Resolution for File-Based Synchronization"](#)

5.10.4.1 Normal Network Synchronization Occurs During File Upload

What happens if a network synchronization from the client was in progress when you started uploading a synchronization file to the mobile server? In this case, the file upload is aborted and a Network Sync In Progress error is reported. Since the data included in the synchronization file is also likely to be within the network synchronization, the appropriate data is synchronized and there is no need to upload the file.

However, if you do upload a synchronization file of data that has already been applied to the mobile server, then the server recognizes that it has already applied one or all of the client transactions. In this case, the mobile server skips the transactions it already has processed, sends any acknowledgements to the client that have not been sent, and applies any client transactions that have not been applied previously.

Additionally, any data destined for the client is sent by the mobile server to the client.

5.10.4.2 Conflict Resolution for File-Based Synchronization

The disconnected nature of file-based synchronization increases the probability that the client may have modified data, which has not been uploaded to the server. Meanwhile, the server may have modified the same data, which creates a conflict. This conflict is solved using the same conflict rules that are configured in the subscription.

5.11 Managing Trace Settings and Trace Files

You can configure the type of tracing that occurs for Data Synchronization components. For more information, see Section 3.2, "Data Synchronization Tracing" in the *Oracle Database Mobile Server Troubleshooting and Tuning Guide*.

5.12 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 5.12.1, "Viewing User Information"](#)
- [Section 5.12.2, "Viewing Publications"](#)
- [Section 5.12.3, "Viewing Publication Items"](#)
- [Section 5.12.4, "Viewing Synchronization Queues"](#)

5.12.1 Viewing User Information

All users that have been added by the administrator (see [Section 4.3, "Managing Users and Groups"](#)) are contained within the mobile server repository.

1. To view information about existing users in the repository, click the "Repository" tab on the "Data Synchronization" home page. The number next to "Users" details the number of users currently in the repository.

As displayed in [Figure 5–20](#), the Repository tab appears.

Figure 5–20 Repository Tab

Page Refreshed Aug 16, 2011 9:50:40 AM

Home Performance Administration **Repository** MGP Platforms

Users and Publications **Queues**

Users (6) In Queue (0)
 Publications (8) Out Queue (0)
 Publication Items (14) Error Queue (0)

Databases Register Database

Deregister

Show All Details | Hide All Details

Select	Details Name	Description	Instance
<input type="radio"/>	Show MAIN	Main Repository for Mobile Server	orcl

Home Performance Administration **Repository** MGP Platforms

- Click "Users", which brings up a list of all users currently in the repository. With this Users screen, you can view everything that is attached to this user, such as application subscriptions, publication items, parameters, SQL scripts, sequences, and performance analysis, see Figure 5–21.

Figure 5–21 Users Repository

Page Refreshed Sep 18, 2014 1:19:10 PM

Search User Go

Subscriptions

Select	User	In Queue Has Data	Out Queue Has Data	Last Synchronization Time
<input checked="" type="radio"/>	S11U1	No	Yes	
<input type="radio"/>	RANDOM	No	Yes	Sep 16, 2014 05:24:41 PM
<input type="radio"/>	JUNE	No	No	
<input type="radio"/>	JOHN	No	No	Sep 03, 2014 04:17:12 PM
<input type="radio"/>	JANE	No	No	
<input type="radio"/>	JACK	No	No	
<input type="radio"/>	BLACK	No	Yes	Sep 12, 2014 12:17:51 PM
<input type="radio"/>	ADMINISTRATOR	No	No	

- 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, and sequences. In addition, if you click the "Consperf Performance Analysis" button, you can generate performance analysis for the publication items. See Section 5.13.3, "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 4.4.1, "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.

5.12.2 Viewing Publications

To view all publications that have been published against the mobile server, click "Publications" under the "Users and Publications" section of the "Data Synchronization" screen, see [Figure 5–22](#). 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, sequences, and users that are attached to this publication.

Figure 5–22 Data Synchronization

The screenshot shows the "Data Synchronization" interface. At the top right, it says "Page Refreshed Sep 18, 2014 1:32:07 PM". Below this is a navigation bar with tabs: Home, Performance, Administration, Repository (selected), MGP, and Platforms. Under the "Users and Publications" section, there are three links: [Users \(7\)](#), [Publications \(7\)](#), and [Publication Items \(10\)](#). Under the "Queues" section, there are three links: [In Queue \(0\)](#), [Out Queue \(7\)](#), and [Error Queue \(0\)](#).

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 2.4.1.7, "Adding Publication Items to Publications" in the *Oracle Database Mobile Server 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 4, "Using Mobile Database Workbench to Create Publications" in the *Oracle Database Mobile Server Developer's Guide*.

5.12.3 Viewing Publication Items

To view publication items, click "Publication Items" under the "Users and Publications" section of the "Data Synchronization" screen, see [Figure 5–22](#). 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 4, "Using Mobile Database Workbench to Create Publications" in the *Oracle Database Mobile Server Developer's Guide*.

5.12.4 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 5.12.4.1, "Viewing Transactions in the In-Queue"](#)
- [Section 5.12.4.2, "Viewing Subscriptions in the Out-Queue"](#)
- [Section 5.12.4.3, "Viewing Server-Side Synchronization Conflicts and Errors in the Error Queue"](#)

5.12.4.1 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.

5.12.4.2 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 5–23](#) displays the "Out-Queue Publications" page.

Figure 5–23 Out-Queue Publications Page

Out Queue Publications

Page Refreshed Jul 28, 2010 1:32:39 PM

Search

1-25 of 52

Select	User	Database	Publication	Complete Refresh Requested
<input checked="" type="radio"/>	CL2LOG_CL1	MAIN	PUB_CL2LOG	No
<input type="radio"/>	CL2LOG_CL1	MAIN	DEFAULT_LINUX	No
<input type="radio"/>	CL2LOG_CL1	MAIN	DEFAULT_EPOC	No
<input type="radio"/>	CL2LOG_CL1	MAIN	DEFAULT_WIN32	No
<input type="radio"/>	CL2LOG_CL1	MAIN	CONSR00T_PUB	No
<input type="radio"/>	CL2LOG_CL1	MAIN	DEFAULT	No
<input type="radio"/>	CL2LOG_CL1	MAIN	DEFAULT_CE	No
<input type="radio"/>	ALBCL	MAIN	ALB_PUB	No
<input type="radio"/>	ALBCL	MAIN	DEFAULT_WIN32	No
<input type="radio"/>	ALBCL	MAIN	C\$MO\$9859	No
<input type="radio"/>	ALBCL	MAIN	DEFAULT_LINUX	No
<input type="radio"/>	ALBCL	MAIN	DEFAULT_EPOC	No
<input type="radio"/>	ALBCL	MAIN	CONSR00T_PUB	No
<input type="radio"/>	ALBCL	MAIN	DEFAULT_CE	No
<input type="radio"/>	ALBCL	MAIN	DEFAULT	No
<input type="radio"/>	USR1	MAIN	DEFAULT	No
<input type="radio"/>	USR1	MAIN	DEFAULT_EPOC	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 5–23](#).

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

Figure 5–24 Publication Items in the Out-Queue Subscription

Publication Items (User: USR1, Publication: DEFAULT)

Page Refreshed Jul 28, 2010 1:34:33 PM

Search

Select	Publication Item	Refresh Mode	Record Count
<input checked="" type="radio"/>	PI_APP_PUB_PROPERTIES	Fast	2
<input type="radio"/>	C\$ALL_SNAPSHOTS	Complete	9

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.

5.12.4.3 Viewing Server-Side Synchronization Conflicts and Errors in the Error Queue

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.
- 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.

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.

If you decide to reapply the records in the transaction to the application tables, click on "Error Queue" off the main page for the mobile server, which displays [Figure 5-25](#).

Note: To modify the error queue programmatically, use the ConsolidatorManager API, as described in Section 2.10, "Resolving Conflicts with Winning Rules" in the *Oracle Database Mobile Server Developer's Guide*.

Figure 5–25 Main Page for Error Queue

Page Refreshed Jul 28, 2010 1:36:24 PM

Search

Select	User	Database	Transaction ID	Errors/Warnings
<input checked="" type="radio"/>	USR3	MAIN	3	1
<input type="radio"/>	USR3	MAIN	6	1
<input type="radio"/>	DATELESS	MAIN	5	10

As [Figure 5–25](#) shows, you can perform the following for the error transactions for each user:

- To view and modify the details for the transaction in the error queue, select the transaction and click "Publication Items". See [Section 5.12.4.3.1, "Modifying Transaction in the Error Queue"](#) for full details.
- If you want to re-execute all records for a user, select the transaction and click "Execute". This may be useful if you have modified the records and want to execute all transactions at once after the modification.
- To delete all transactions for a user from the error queue, select the transaction and click "Purge".
- To search for a user error queue transaction, enter the user name in the "Search" field and click "Go". Based on your search criteria, the mobile server displays the result.

5.12.4.3.1 Modifying Transaction in the Error Queue Before re-executing the transaction, correct the error within each record of the transaction, as follows:

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.
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 5–26](#). 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 did not cause any problems.

Figure 5–26 Error Queue Records for a Single Publication Item

Page Refreshed Jul 28, 2010 1:40:23 PM

Search

Select All | Select None

Select	Details	DML Operation	Version	Message	Primary Key ID	First 1 Fields STATUS
<input type="checkbox"/>	Show	Update	1	Update/Delete Error :: No Data Found	1	1

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 5–27](#).

Figure 5–27 Show Details of Error Queue Record

Page Refreshed Jul 28, 2010 1:41:59 PM

Search ID Go

Update DML Insert Update Field ID Go

Select All | Select None

Select	Details	DML Operation	Version	Message	Primary Key ID	First 1 Fields STATUS
<input type="checkbox"/>	<input type="button" value="Hide"/>	Update	1	Update/Delete Error :: No Data Found	1	1

Record Details

Update DML Update

Field Name	Data Type	Server Record	Error Queue Record
<ID>	NUMBER(8,0)		1
STATUS	NUMBER(5,0)		1
INS_DATE(YYYY-MM-DD HH24:MI:SS)	DATE		2010-07-13 19:15:34

* Field names inside <> are primary key fields

3. Correct the reason why the error occurred in the first place. By performing one or more of the following, you are modifying the record within the error queue. The changes that you make in this record are not executed until you select the transaction and click the "Execute" button in the main Error Queue screen shown in Figure 5–25.

- **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.

- **Edit Fields and Select DML Action:** Modify any fields in the Error Queue Record section under Record Details. Once you have completed all necessary changes to any records, then verify the DML action to be executed for this record in the "Select DML Action" pulldown. You can select Update, Insert or Delete in the Update DML pull-down. Clicking "Save" modifies the record in the transaction in the error queue.

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. This is shown in Figure 5–26.

After selecting the desired rows, choose the buttons at the top of the page or performs 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 one or more rows with the checkboxes and click "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-execute a subset of the original records. After you create a new transaction, you can execute it off the main Error Queue page.

5.13 Monitoring and Analyzing Performance

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

- [Section 5.13.1, "Viewing Sync Server Statistics"](#)
- [Section 5.13.2, "Displaying MGP Cycles and Statistics"](#)
- [Section 5.13.3, "Analyzing Performance of Publications With the Consperf Utility"](#)
- [Section 5.13.4, "Monitoring Synchronization Using SQL Scripts"](#)

5.13.1 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 5–28](#), you can see the 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 5–28 Performance Page

Home	Performance	Administration	Repository	MGP	Platforms
General Synchronization History Sessions Synchronization Statistics			Conspert To do performance analysis using the consper utility, start with the Users table and choose a subscription upon which you can perform consper analysis.		
Active Sessions Statistics			Last 24 Hours Sessions Statistics		
Summary			Summary		
Session Count 0			Session Count 1		
Upload Phase Sessions 0			Average Duration (seconds) 5		
Download Phase Sessions 0			Maximum Duration (seconds) 5		
Average Duration (seconds) 0			Average Record Count 38		
Maximum Duration (seconds) 0			Maximum Record Count 38		
Average Record Count 0			Total Record Count 38		
Total Record Count 0			Average Byte Count 1652		
Average Byte Count 0			Maximum Byte Count 1652		
Total Byte Count 0			Total Byte Count 1652		
Upload Phase			Upload Phase		
Average Duration (seconds) 0			Average Duration (seconds) 0		
Maximum Duration (seconds) 0			Maximum Duration (seconds) 0		
Average Record Count 0			Average Record Count 0		
Maximum Record Count 0			Maximum Record Count 0		
Total Record Count 0			Total Record Count 0		
Average Byte Count 0			Average Byte Count 346		
Maximum Byte Count 0			Maximum Byte Count 346		
Total Byte Count 0			Total Byte Count 346		
Download Phase			Download Phase		
Average Duration (seconds) 0			Average Duration (seconds) 5		
Maximum Duration (seconds) 0			Maximum Duration (seconds) 5		
Average Record Count 0			Average Record Count 38		
Maximum Record Count 0			Maximum Record Count 38		
Total Record Count 0			Total Record Count 38		
Average Byte Count 0			Average Byte Count 1306		
Maximum Byte Count 0			Maximum Byte Count 1306		
Total Byte Count 0			Total Byte Count 1306		

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.

5.13.2 Displaying MGP Cycles and Statistics

As shown in the Mobile Manager Home page on [Figure 5–29](#), the status for all MGP Jobs are listed in a table under the Data Synchronization section on the top. To see the statistics for each MGP Job, select the link under the Status column for the desired MGP job. For more details, refer to [Section 6.3.4, "Viewing MGP Cycle Statistics"](#).

Figure 5–29 MGP Status

The screenshot displays the Oracle Database Mobile Server 12c Mobile Manager interface. At the top, it shows the Oracle logo and 'Database Mobile Server 12c Mobile Manager'. There are 'Logout' and 'Help' buttons. A navigation bar includes 'Mobile Servers | Mobile Devices' and 'Mobile Server: bdbcn6:9091'. Below this is a menu with 'Home', 'Applications', 'Users', and 'Administration'. A status bar indicates 'Page Refreshed Sep 18, 2014 1:41:02 PM'.

The main content area is divided into two sections: 'General' and 'Data Synchronization'.

General

Status	Up
Version	12.1.0.0.0
Up Since	Sep 16, 2014 10:05:20 AM
Mode	GlassFish Server

Data Synchronization

MGP Status	Database	Status
	MAIN	Idle

Queues: [In Queue \(0\)](#), [Out Queue \(7\)](#), [Error Queue \(0\)](#)

Main Database

Database Version	12.1.0.1.0
JDBC URL	jdbc:oracle:thin:@//bdbcn16.cn.oracle.com:1522/pdborcl.cn.oracle.com
JDBC Driver	Oracle JDBC driver
JDBC Version	11.2.0.1.0
Schema Name	DMS121_GFS4_0904

5.13.3 Analyzing Performance of Publications With the Conspert Utility

The Conspert 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 Conspert 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 Conspert tool. We either modify the query template or type of logical delete or insert for you or enable you to 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 Conspert 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 Conspert utility, see Section 1.2.1 "Analyzing Performance of Publications With the Conspert Utility" in the *Oracle Database Mobile Server Troubleshooting and Tuning Guide*. The Conspert tool uses the Oracle Database

version of the Explain Plan; thus, for full details on the Oracle Database Explain Plan, refer to the Oracle Database documentation.

5.13.4 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 Mobile Server Troubleshooting and Tuning Guide*.

Managing Jobs with the Job Scheduler

You can schedule one or more jobs to execute at a specific time. To enhance performance, you can manage your jobs across one or more Job Schedulers. Each mobile server in the farm starts a Job Scheduler, also known as a Job engine.

By default, jobs are not registered for specific Job Schedulers. Instead, all Job Schedulers distribute and execute the registered jobs. All Job Schedulers work in tandem with each other, providing failover to each other. Only one job can execute at a Job engine at a time.

You can further manage your jobs by selecting which Job Scheduler on a specified mobile server is to execute the job. Or you can start a Standalone Job engine to execute jobs, where a Standalone Job engine exists outside of the mobile servers.

The following sections describe how you can schedule jobs through the Mobile Manager and manage one or more job engines.

- [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 or Creating Jobs Using ConsolidatorManager APIs"](#)

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 6.4, "Managing or Creating Jobs Using ConsolidatorManager APIs"](#).

The Job engine that monitors the job execution is the Job Scheduler. For example, by default, the Job Scheduler fires off the default 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.

Each mobile server in the farm starts a Job Scheduler to enhance the performance of your job execution through failover and executing jobs concurrently. You can also start a Standalone Job engine, as described in Chapter 6, "Create and Manage Jobs With APIs" in the *Oracle Database Mobile Server Developer's Guide*.

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

6.2 Managing the Job Engine

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

Note: If you schedule jobs at any of the Job Schedulers, they will be managed across all Job Schedulers in the farm unless you designate the job to be executed on a specified mobile server.

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

Job Scheduler

Page Refreshed Aug 16, 2011 10:02:47 AM

[Home](#) [Administration](#)

General



Status **Up**
 Status Days **0.93**
 Status Date **Aug 15, 2011 11:34:51 AM**
 Job History [6](#)
 Related Links [MGP](#)
[Data Synchronization](#)

Alerts

Select	Name	Severity	Alert Triggered
	(No items found)		

Active Jobs

Name	Class Name	Param Value	Start Time	Duration (seconds)
(No items found)				

[Home](#) [Administration](#)

To stop the Job Scheduler, click "Stop". Stopping the Job Scheduler prevents any new jobs from starting on this mobile server. However, any existing jobs on this mobile server will continue to execute until finished. Stopping the Job Scheduler does not kill any existing jobs. All other jobs are moved to other available Job Schedulers in the farm.

If you want to prevent a single job from being launched, disable the application on the Administration screen. See [Section 6.3.5, "Enabling or Disabling Jobs"](#) for more information on disabling applications.

6.2.2 Checking Job Scheduler Alerts

When the Job Scheduler fails, then the Alerts table displays these exceptions as critical alerts. When the Job Scheduler 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. All registered jobs from all Job

Schedulers are listed. 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

Job History Page Refreshed Jul 29, 2010 12:49:07 PM

Search

Job Properties
 Name:
 Class Name:
 Result: All

From
 Date: 7/22/10
 Example: 10/31/03
 Time: 0 45 AM PM
 Time Zone: Pacific Standard Time

To
 Date: 8/5/10
 Example: 10/31/03
 Time: 0 45 AM PM

Results

Select	ID	Name	Class Name	Result	Finish Time	Duration (seconds)	Message
<input checked="" type="radio"/>	196	PURGE_HISTORY_DEFAULT	oracle.lite.sync.PurgeHistoryJob	✓	7/29/10 12:00 AM	1	Job parameter = "History=Sync,MGP,Job; Days=7". purgeSyncHistoryPubltems=0 purgeSyncHistory=0 purgeMgpHistoryClients=0 purgeMgpHistoryLogTables=0 purgeMgpHistory=0 purgeJobHistory=1

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, "Viewing MGP Current Cycle Statistics"
- Section 6.3.4, "Viewing MGP Cycle Statistics"
- Section 6.3.5, "Enabling or Disabling Jobs"

- [Section 6.3.6, "Deleting Jobs"](#)
- [Section 6.3.7, "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.

In the Preferred Location field, specify a preferred Job Scheduler location. This enables you to specify that the job executes on a specific mobile server or, if available, a Standalone Job engine. All available Job Schedulers and the Standalone Job engine are shown in the drop-down list.

If you do not specify a preferred location, the job is executed by one of the available Job Schedulers in the registered mobile servers or Standalone Job Scheduler, if any. This job can fail over to any of the Job Schedulers, if necessary. However, if a job has been given a preference for specific Job Scheduler, it will keep running on that Job Scheduler as long as the Job Scheduler is available. Only if that Job Scheduler becomes unavailable will it failover to another Job Scheduler in the farm.

Note: You can start a Standalone Job engine if all the mobile servers are busy with processing and you want to execute one or more jobs in a separate process for performance reasons.

A Standalone Job engine can be started with the ConsolidatorManager API, as described in Chapter 6, "Create and Manage Jobs With APIs" in the *Oracle Database Mobile Server Developer's Guide*.

Figure 6–3 Create a New Job - Top Section

The screenshot shows a web form titled "Job" with two main sections: "General" and "Job Class".

- General Section:**
 - Job Name: [Text Input]
 - Enabled:
 - Save to Job History:
 - Preferred Location: [Dropdown Menu]
- Job Class Section:**
 - MGP: (Selected)
 - Custom:
 - Apply/Compose Mode: [Dropdown Menu] (Apply Only)
 - Database: [Dropdown Menu] (MAIN)
 - Class Name: [Text Input]
 - Parameter Value: [Text Input]

Under the "Job Class" section, select if the Job class is for an MGP process or another class.

- **MGP process:** For an MGP process, select if this is for "Apply Only" or "Apply and Compose". The MGP process can be modified to perform application only 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.

For the "Select Database" pull-down, select the database where the MGP job process shall execute.

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.

- Custom class:** 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 6.3.7, "Default Jobs"](#).

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

Schedule

Time Zone **Pacific Standard Time**

Start

Immediately
 Later

Date

Example: 10/31/03

Time AM PM

Expiration

Never
 Expire
 Expire

Limit (minutes)

Cancel if not started within the time limit

Repeat

One
 Time Only
 Interval

Frequency (seconds)

Weekly
 Frequency (weeks)

Days of Week
 Mo Tu We Th Fr Sa Su

Monthly
 Frequency (months)

Days of Month
 1 2 3 4 5 6 7
 8 9 10 11 12 13 14
 15 16 17 18 19 20 21
 22 23 24 25 26 27 28
 29 30 31 LAST

Repeat Until

Indefinite
 Custom

Date

Example: 10/31/03

Time AM PM

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 Viewing MGP Current Cycle Statistics

As shown in the Mobile Manager Home page, the status for all MGP Jobs are listed in the "MGP Status" table under the "Data Synchronization" section on the top. To see the statistics for each MGP Job, select the link under the "Status" column for the desired MGP Job.

Figure 6–5 MGP Status

Home Applications Users Administration

General

 Status 1 of 2 up
Version 12.1.0.0.0
Up Since Nov 6, 2014 4:17:17 PM
Mode GlassFish Server

Data Synchronization

MGP Status	Database	Status
	MAIN	Idle

Queues [In Queue \(0\)](#)
[Out Queue \(7\)](#)
[Error Queue \(0\)](#)

Main Database

Database Version 12.1.0.1.0
JDBC URL jdbc:oracle:thin:@//bdbcn16.cn.oracle.com:1522/pdborcl.cn.oracle.com
JDBC Driver Oracle JDBC driver
JDBC Version 11.2.0.1.0
Schema Name DMS121_GFS4_0904

On the MGP Current Cycle page as shown in Figure 6–6, the administrator can view the current cycle status of the desired MGP Job.

Figure 6–6 MGP Current Cycle

MGP Current Cycle Page Refreshed Sep 18, 2009 11:20:57 PM

Summary

Cycle ID	Apply Record Count
Database	Apply Duration (seconds)
Type	Process Log Record Count
Phase	Process Log Duration (seconds)
Result	Compose Record Count
Start Time	Compose Duration (seconds)
Finish Time	
Duration (seconds)	

Log Tables Processed

Search

Table Name	Dirty Record Count
(No items found)	

Shared Publication Items Processed

Search

Select Publication Item	Group	Compose Duration (seconds)	Representing User
(No items found)			

Users Processed

Search

User	Apply Record Count	Apply Pub Item Count	Has Apply Conflicts	Has Apply Errors	Apply Duration (seconds)	Compose Record Count	Compose Pub Item Count	Has Compose Errors	Compose Duration (seconds)
(No items found)									

6.3.4 Viewing MGP Cycle Statistics

From the mobile server home page, select "Data Synchronization" and navigate to the "MGP" tab to display more MGP Job statistics. The mobile server administrator can view MGP job statistics as shown in Figure 6–7. 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 6–7 MGP Page

General

[Job Scheduler\(Up\)](#) [MGP Current Cycle](#) [MGP Apply/Compose Cycles](#) [MGP Apply/Compose Cycle Statistics](#)

Last 24 Hours MGP Apply/Compose Cycle Statistics

Summary		Apply Phase		Process Log Phase		Compose Phase	
Cycle Count	0	Cycle Count	0	Cycle Count	0	Cycle Count	0
Average Duration (seconds)	0	Average Duration (seconds)	0	Average Duration (seconds)	0	Average Duration (seconds)	0
Maximum Duration (seconds)	0	Maximum Duration (seconds)	0	Maximum Duration (seconds)	0	Maximum Duration (seconds)	0
Average Record Count	0	Average Record Count	0	Average Record Count	0	Average Record Count	0
Maximum Record Count	0	Maximum Record Count	0	Maximum Record Count	0	Maximum Record Count	0
Total Record Count	0	Total Record Count	0	Total Record Count	0	Total Record Count	0

[Home](#) [Performance](#) [Administration](#) [Repository](#) **MGP**

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 5.6, "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.

6.3.5 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.

6.3.6 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.

6.3.7 Default Jobs

Oracle Database Mobile Server 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

6.3.7.1 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.

6.3.7.2 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 or Creating Jobs Using ConsolidatorManager APIs

Application developers can create and manage their jobs using the ConsolidatorManager APIs. For full details, see Chapter 6, "Create and Manage Jobs With APIs" in the *Oracle Database Mobile Server Developer's Guide*.

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:

- [Section 7.1, "Customize the Mobile Client Software Installation for Your Mobile Device"](#)
- [Section 7.2, "Configuring Mobile Clients Before Installation"](#)
- [Section 7.3, "Managing Devices"](#)
- [Section 7.4, "Configuring and Customizing Your Mobile Device Platform"](#)
- [Section 7.5, "Sending Commands to Your Mobile Devices"](#)
- [Section 7.6, "Managing Device Software Updates"](#)
- [Section 7.7, "Using the Device Manager Agent \(dmagent\) on the Client"](#)
- [Section 7.8, "Managing the Network Protocol Between the Device and the Mobile Client Software"](#)
- [Section 7.9, "Installation Configuration \(INF\) File"](#)
- [Section 7.10, "Defining Device Manager Commands With the Device Manager OTL Tag Language"](#)
- [Section 7.11, "Using Mobile Manager to Manage iOS Devices"](#)

7.1 Customize the Mobile Client Software Installation for Your Mobile Device

The mobile device software for your language and platform is installed when you install the mobile client on your device.

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 7.4, "Configuring and Customizing Your Mobile Device Platform"](#) for more information.

Once installed, you can locally or remotely manage the client using the Device Manager tool. See [Section 7.7, "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 `ose.ini`, `devmgr.ini` and `odbc.ini` files. However, you can pre-configure some of the parameters destined for the client `ose.ini`, `devmgr.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 mobile client 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 7.9, "Installation Configuration \(INF\) File"](#).

Note: The `ose.ini`, `devmgr.ini` and `odbc.ini` files are available in under `$MOBILE_CLIENT_HOME/bin`. 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 when directed by 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"](#)

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:

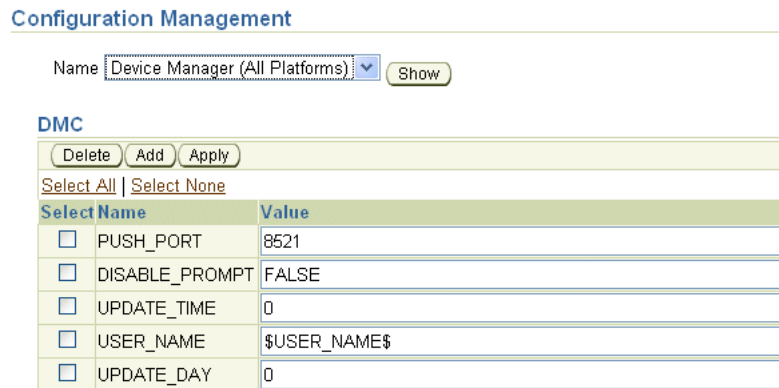
```
<ini>
<item name="DEVGR" section="DMC">
<item name='PUSH_PORT'>8521</item>
<item name='DISABLE_PROMPT'>FALSE</item>
<item name='UPDATE_DAY'>0</item>
<item name='UPDATE_TIME'>0</item>
</item>
</ini>
```

When you select Device Management->Administration->Configuration Management, you can modify the device management parameters as follows:

- Modify generic platform parameters, as shown in Figure 7-1.
- Modify platform-specific parameters, as shown in Figure 7-2. These parameters exist only for certain platform types.

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



Where the parameters are as follows:

Table 7-1 Device Management Parameters in DEVGR.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 <code>devmgr.ini</code> file.

Table 7-1 (Cont.) Device Management Parameters in DEVMgr.INF File

Parameter	Description
DISABLE_PROMPT	<p>The DISABLE_PROMPT parameter accepts a TRUE or FALSE value, which causes the following action:</p> <ul style="list-style-type: none"> ■ 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	<p>Day when the mobile device checks for software updates. Used in combination with UPDATE_TIME. UPDATE_DAY takes 0 - 8 which translates to the following days:</p> <ul style="list-style-type: none"> ■ Never = 0 ■ Daily = 1 ■ Sunday = 2 ■ Monday = 3 ■ Tuesday = 4 ■ Wednesday = 5 ■ Thursday = 6 ■ Friday = 7 ■ Saturday = 8
UPDATE_TIME	<p>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:</p> <ul style="list-style-type: none"> ■ 00:00 = 0 ■ 01:00 = 1 ■ 12:00 = 12 ■ 13:00 = 13 ■ 23:00 = 23
USER_NAME	<p>Do not modify; automatically retrieves the user name from the mobile server when downloaded to the client.</p>

Note: You can also modify the UPDATE_DAY and UPDATE_TIME parameters on the client through the dmagent UI. See [Section 7.7, "Using the Device Manager Agent \(dmagent\) on the Client"](#) for details.

Figure 7-2 Platform-Specific Parameters

Configuration Management

Name

DEFAULT

|

Select Name	Value
<input type="checkbox"/> SQLITE.DATA_DIRECTORY	\$_APP_DIR\$\sqlite\sqlite_db

7.3 Managing Devices

From the Mobile Devices screen, as shown in [Figure 7-3](#), you can perform the following:

- **Add**—Click Add to add a new device.
- **View**—Select the device to view information about this device.
- **Delete**—Select the checkbox next to the desired device and click **Delete**.

Note: If you want to delete a device, use the `delete` method from the `Device` class. To retrieve the `Device` object, use either the `getDevice` or `getDeviceByName` methods, as demonstrated below.

If the device id is available, the following can be directly used:

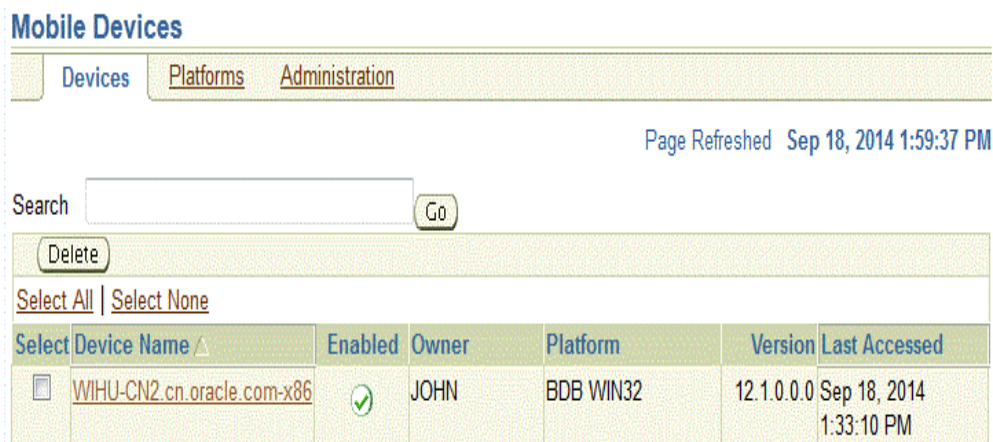
```
if (oracle.lite.resource.ResourceManager.getInstance() == NULL)
oracle.lite.resource.ResourceManager.initialize(JDBC_URL, USER,
PASSWORD);

oracle.lite.resource.Device d =
oracle.lite.resource.ResourceManager.getInstance().getDevice(device
Id);

d.delete();
```

If the device id is not available, then you can provide the device name, which is shown on the Mobile Manager UI in the `oracle.lite.resource.User.getDeviceByName (deviceName)` method. Once retrieved, use the `delete` method of the `Device` object as demonstrated above.

Figure 7–3 Devices Page



Once you select the intended device, several tabs at the top provides 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 cover each of these tabs.

- [Section 7.3.1, "Configuring the Mobile Device through Properties"](#)
- [Section 7.3.2, "Viewing Device Information"](#)
- [Section 7.3.3, "Viewing Database Information"](#)
- [Section 7.3.4, "Viewing Software Information"](#)
- [Section 7.3.5, "Commands"](#)
- [Section 7.3.6, "Queue"](#)
- [Section 7.3.7, "Command History"](#)

7.3.1 Configuring the Mobile Device through Properties

To modify the configuration of a specific device, select the device. [Figure 7–4](#) is displayed:

Figure 7–4 Mobile Device Properties and Information

Device

Properties Device Info Database Info Software Info Commands Queue Command History Device Logs

Page Refreshed Aug 16, 2011 10:12:48 AM

General

Device Name

Enabled

Upgradable

ID **81**

Valid **Yes**

Type **WIN32_x86_US_SQLite**

Last Accessed **Aug 15, 2011 6:29:03 PM**

Access Count **503**

Address

Address

Network Provider

Properties Device Info Database Info Software Info Commands Queue Command History Device Logs

To modify the configuration of a specific device, 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 7.4.2, "Enabling or Disabling All Mobile Devices in a Platform"](#) or [Section 7.3.1.1, "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, "Managing Device Software Updates"](#) for more information.
4. Valid: If the Device Manager software is installed and correct. If de-installed, Valid displays "No".
5. 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.
6. 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, SMS, or RAPI. For Android client, the only supported network provider is SMS. If you have added another network provider, these custom network providers will also be included in the list. For iOS client, there is no need to choose the network provider type.

Note: To create a network provider, see [Section 7.8, "Managing the Network Protocol Between the Device and the Mobile Client Software"](#).

7.3.1.1 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 7.3.1, "Configuring the Mobile Device through Properties"](#) 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 7.5, "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.

7.3.2 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 is 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. 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.

7.3.3 Viewing Database Information

When you select the Database Info tab, you see all of the information about any databases 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 third section displays the configuration in raw form in the `DEVMGR.INI` files on the client. Each section in the `DEVMGR.INI` file is displayed. The purpose of this section is for you to view the sections and parameters in the INI files.

7.3.4 Viewing Software Information

You can view all of the 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.

7.3.5 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 7.5, "Sending Commands to Your Mobile Devices"](#).

7.3.6 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.

7.3.7 Command History

This shows all of the commands executed against this device.

7.4 Configuring and Customizing Your Mobile Device Platform

Oracle Database Mobile Server ships with a number of predefined 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.

A mobile 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 4.3.5, "Mobile Client Install" in the *Oracle Database Mobile Server Installation 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 7.4.1, "Modifying Platform Properties for Installation"](#)

- [Section 7.4.2, "Enabling or Disabling All Mobile Devices in a Platform"](#)
- [Section 7.4.3, "Extend or Create a Custom Platform"](#)

7.4.1 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, 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 a user-provided name. Once extended, modify this platform with your own characteristics. For instructions on how to extend one of the provided platforms, see [Section 7.4.3, "Extend or Create a Custom Platform"](#).

To modify how the platform is installed, perform 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 `$MOBILE_HOME/Mobile/Server/admin/repository/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 7.9, "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 7.4.2, "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 7.6, "Managing Device Software Updates"](#) for more information on updating software on your device.

7.4.2 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 7.3.1.1, "Enabling or Disabling a Mobile Device"](#). However, if you want to enable or disable all devices for a platform, then see [Section 7.4.1, "Modifying Platform Properties for Installation"](#).

When disabling all devices for a platform, you could send a deinstall command (see [Section 7.5, "Sending Commands to Your Mobile Devices"](#)) and then disable all of the analog mobile devices.

7.4.3 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 7.4.3.1, "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 with specific instructions to modify the client machine. See [Section 7.4.3.2, "Create a Custom Platform By Extending an Existing Platform"](#) for more information.

7.4.3.1 Enable a Platform for Your Mobile Client

Not all of the possible platforms are enabled for 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.

7.4.3.2 Create a Custom Platform By Extending an Existing Platform

You may wish to install additional binaries, applications, or provide specific instructions to modify the client machine when the client platform is installed on 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 `$MOBILE_HOME/Mobile/Server/admin/repository/setup/dmc/` to where the setup INF files are located. Create an empty INF file for your new platform. As discussed in [Section 7.9, "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 7.9, "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 Sending Commands to Your Mobile Devices

As an administrator, you can create and send commands to any mobile device. The following sections describe how to send existing commands to devices and create new commands for your own purposes:

- [Section 7.5.1, "Scheduling or Sending Commands"](#)
- [Section 7.5.2, "Modifying Existing Commands"](#)
- [Section 7.5.3, "Creating New Commands"](#)
- [Section 7.5.4, "Creating Group Commands"](#)
- [Section 7.5.5, "Enabling or Disabling Mobile Device Commands"](#)
- [Section 7.5.6, "Viewing the Mobile Device Command History"](#)

7.5.1 Scheduling or Sending Commands

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

7.5.1.1 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 Windows Mobile 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 7.5.3, "Creating New Commands"](#).

7.5.1.1.1 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 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 platform.
 - The user must have access to the application.

If the application already is installed on the client and an update is available, this command forces an update for the client.

- Retrieve Software Information—Retrieves information regarding Oracle Database Mobile Server and its 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 Manager 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'. 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 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 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 MOBILE_HOME\Mobile\Server\admin\repository\devmgr\- Modify Configuration—Modify configuration settings in DEVPMGR.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 script directory.
- De-install Oracle Database Mobile Server—Remotely de-install Oracle Database Mobile Server on the client device.
- 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.

7.5.1.2 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 7-5](#), 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.

Note: You can only use fast refresh with a high priority restricting predicate. If you use any other type of refresh, the high priority restricting predicate is ignored.

- 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.

Figure 7-5 The General Section of the Command Scheduler

Schedule Command

General

General		Parameter	
Name	1313515179638	Command error	
Description	1313515179638	Send	High
<input checked="" type="checkbox"/> Enabled		Priority	High
<input type="checkbox"/> Save to History		Extra	
		Param	

- As Figure 7-6 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 7-6 Timing Section of the Command Scheduler

Schedule

Start		Expire	
<input checked="" type="radio"/> Immediately		<input checked="" type="radio"/> Never	
<input type="radio"/> Later		<input type="radio"/> Expire	
		<input type="radio"/> Expire	
Time	10 15 AM PM	Limit (Minutes)	60
Date	8/16/11	<small>Cancel if not started within the time limit</small>	
Repeat		Repeat Until	
<input checked="" type="radio"/> One Time Only		<input checked="" type="radio"/> Indefinite	
<input type="radio"/> Interval	Frequency (Seconds) 60	<input type="radio"/> Custom	
<input type="radio"/> Weekly	Frequency (Weeks) 1	Date	8/16/11
	Day of Week	Time	10 15 AM PM
	<input type="checkbox"/> Mo <input type="checkbox"/> Tu <input type="checkbox"/> We <input type="checkbox"/> Th <input type="checkbox"/> Fr <input type="checkbox"/> Sa <input type="checkbox"/> Su		
<input type="radio"/> Monthly	Frequency (Months) 1		
	Day of Month		
	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5 <input type="checkbox"/> 6 <input type="checkbox"/> 7		
	<input type="checkbox"/> 8 <input type="checkbox"/> 9 <input type="checkbox"/> 10 <input type="checkbox"/> 11 <input type="checkbox"/> 12 <input type="checkbox"/> 13 <input type="checkbox"/> 14		
	<input type="checkbox"/> 15 <input type="checkbox"/> 16 <input type="checkbox"/> 17 <input type="checkbox"/> 18 <input type="checkbox"/> 19 <input type="checkbox"/> 20 <input type="checkbox"/> 21		
	<input type="checkbox"/> 22 <input type="checkbox"/> 23 <input type="checkbox"/> 24 <input type="checkbox"/> 25 <input type="checkbox"/> 26 <input type="checkbox"/> 27 <input type="checkbox"/> 28		
	<input type="checkbox"/> 29 <input type="checkbox"/> 30 <input type="checkbox"/> 31 <input type="checkbox"/> Last		

Revert Apply

- Click "Apply". The Mobile Manager displays a confirmation message.

7.5.2 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 7-7](#) displays, the "Command Management" page appears.

Figure 7-7 Command Management Page

Select Name	Description	Command
<input type="checkbox"/> AndroidBootup	Retrieve Android device information	DevInfo, ClientConfig
<input type="checkbox"/> DSN	ODBC DSN information	ODBC_INFO
<input type="checkbox"/> DbInfo	Database information	DB_INFO
<input type="checkbox"/> De-Install	De-Install Oracle Mobile client	uninst
<input type="checkbox"/> DevInfo	Device information	DEV_INFO
<input type="checkbox"/> DeviceInfo	Retrieve device information	DevInfo, DbInfo, DSN, ClientConfig, NTFS
<input type="checkbox"/> Install	Install application	\$setup
<input type="checkbox"/> ModifyINI	Modify configuration	UPDT_CONF
<input type="checkbox"/> NTFS	Notification	NTFS
<input type="checkbox"/> ClientConfig	Oracle Mobile client configuration	CONFIG_INFO
<input type="checkbox"/> ResetPassword	Reset password	LOGON
<input type="checkbox"/> SoftwareInfo	Retrieve software information	\$solver?mode=i&name=\$
<input type="checkbox"/> StopDMC	Stop device management client	exit
<input type="checkbox"/> SyncAndDelete	Synchronize and delete databases	SYNC_DELETE
<input type="checkbox"/> SyncLog	Retrieve synchronization log	SYNC_LOG
<input type="checkbox"/> Synchronize	Synchronize databases	SYNC
<input type="checkbox"/> UpdateDMC	Update device management client	\$setup?mode=u
<input type="checkbox"/> UploadFile	Retrieve a file from the device	upload
<input type="checkbox"/> ValidateDB	Validate database	VALIDATE

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

- Click the required "Command Name" link. The "Properties" page for this command appears.
- Enter the command name, description, and syntax in the corresponding fields. For more information on modifying these fields, see [Section 7.5.3, "Creating New Commands"](#).
- To check the accuracy of the command syntax, click "Syntax Check" button. If no errors are found, the Mobile Manager displays a confirmation message.
- Click "Apply". The Mobile Manager displays a confirmation message.

7.5.2.1 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.

You can only use fast refresh with a high priority restricting predicate. If you use any other type of refresh, the high priority restricting predicate is ignored.

- 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.

7.5.3 Creating New Commands

You can create commands using the OTL scripting language, as described in [Section 7.10, "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 `MOBILE_HOME\Mobile\Server\admin\repository\setup\dmc\otl` directory.
2. With any editor, add all OTL commands that you want executed within the file. See [Section 7.10, "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. If you ask the user to enter parameters, then add the parameter definitions. See [Section 7.5.2.1, "Adding Parameters to Mobile Device Commands"](#) for a full description.
11. Click "Apply".
12. You have now successfully created a new command. After you send the command to the device, you can execute this command against your mobile device. See [Section 7.5.1, "Scheduling or Sending Commands"](#) for information on how to send the command to the device.

7.5.4 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.

7.5.5 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".

7.5.6 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".

7.6 Managing Device Software Updates

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

- [Section 7.6.1, "Configuring the Device to Receive Required Software Updates"](#)
- [Section 7.6.2, "Configuring Application Software for Automatic Update"](#)
- [Section 7.6.3, "Initiate Updates of Oracle Database Mobile Server Software from the Client"](#)

7.6.1 Configuring the Device to Receive Required Software Updates

If you configure for automatic software updates, then when a new software update comes available—either for the mobile client software or for any applications installed on the client—the mobile device will receive these updates. However, if you want a device to stay with the level of software that is currently installed, disallow automatic updates.

There are two types of software updates that you can control, as detailed in the following sections:

- [Section 7.6.1.1, "Allowing Automatic Software Updates for the Oracle Database Mobile Server Platform"](#)
- [Section 7.6.1.2, "Updating Application Software On Each Client"](#)

7.6.1.1 Allowing Automatic Software Updates for the Oracle Database Mobile Server Platform

You can configure to automatically allow software updates for the devices through one of two methods:

- Set `Upgradable` to Yes/No—The `Upgradable` field enables you to configure for automatic software updates, so that 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, set `Upgradable` to No.

See [Section 7.3.1, "Configuring the Mobile Device through Properties"](#) for details on how to modify the `Upgradable` field within the Mobile Manager GUI.

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

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 platforms.
 3. Click the appropriate client "WIN32" platform to display its properties.
 4. Change the value of Upgradable to Yes/No. Click "OK".
- Set 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 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");
```

7.6.1.2 Updating Application Software On Each Client

You can control whether a new version of an application software is downloaded on each client and which users receive the latest update. The default configuration is for all devices attached to a user to 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 SQLite Win32 version 11.2.0.0.0; however, you want Tom's devices to upgrade to the new version, which is SQLite Win32 version 11.3.0.0.0. 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`.

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, which is denoted by the version number. Any modification of the version in the first or second position is a major update. For example, any version that changes from 10.0.0 to 10.1.0 or 11.0 is a major update. This is the default.
 - Minor—The devices attached to this user receives only minor software updates. This includes only patch releases. For example, if the client software is version 11.0.0, then modifications only apply to the third position or later constitutes a patch update. This would include the version numbers 11.0.1 or 11.0.0.1. It would not include the 11.1.0 which would be a major update.
 - 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. For example, if the version shows a major release, then this device receives the update. This is the default.
- **Minor**—The devices attached to this user receives only minor software updates. For example, if the version shows only a minor patch release, then this device receives the update. However, if the software released is a major version upgrade, then this is not applied.
- **False**—The devices attached to this user does not receive any software updates.

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

7.6.1.3 Rolling Out Updates With Controlled Upgrade

If you want to roll out software updates in a staggered fashion to a subset of your users at a time, you can use the controlled update. A controlled update enables the following:

- You can deploy the latest software or patches to a set of users to limit the impact of your IT team handling multiple responses from affected users.
- You can deploy the latest software or patches to a set of users for testing purposes, while keeping the majority of the users on an older version of the software.

To perform a controlled update, perform the following within Mobile Manager:

1. Create a group with all of the users that you want to receive the update.
2. Edit the group and set the update policy for the group to All, Major, Minor or Disable.

Alternatively, you can use the APIs and perform the following:

1. Create a group with the `MobileResourceManager.createGroup` method.
2. Set the group update policy with the following method, where the value can be "major," "minor," or "false."

```
group.setPolicy (ResourceConst.UPDATE_SOFTWARE, "major");
```

3. Add all users to be within the group with the `MobileResourceManager.addUsersToGroup` method.

7.6.2 Configuring Application Software for Automatic Update

In order for the mobile server and the device to know if an application software is to be updated, you need to configure the INF file to detail the current version. This version is checked against what is currently installed on the client, which then enables the mobile server and the device management to decide whether an automatic software update is necessary.

The following sections describe how to configure your application software for automatic update:

- [Section 7.6.2.1, "Configuring Major Software Updates for Download"](#)
- [Section 7.6.2.2, "Configuring Patches or Minor Updates for Download"](#)

7.6.2.1 Configuring Major Software Updates for Download

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. Initiate a synchronization. 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.

Alternatively, the client user can invoke `update.exe` to check for the latest version of the software. See [Section 7.6.3, "Initiate Updates of Oracle Database Mobile Server Software from the Client"](#) for more details on the update tool.

7.6.2.2 Configuring Patches or Minor Updates for Download

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\Mobile\Server\admin\repository\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>
```



```

        </item>
    </file>
</install>
</setup>

```

Note: For a full description of INF files and the elements within them, see [Section 7.9, "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 (1.2.3) tells DMS that the patch is meant for the application version 1.2.3.
- The INF file must have an ID, which is used for determining patch dependencies. This ID can be any number you choose.

If you have dependencies among patches, use the `dependency` element to indicate these dependencies. For example, the previous patch for `o1obj40.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 `o1obj40.dll` by configuring the ID number of 1001 in the `dependency` element.

```

<setup name="SQLite WIN32" version="10.0.0.0" id='1002'>
  <property>
    <dependency>1001</dependency>
  </property>
</setup>

```

Update the patch, as follows:

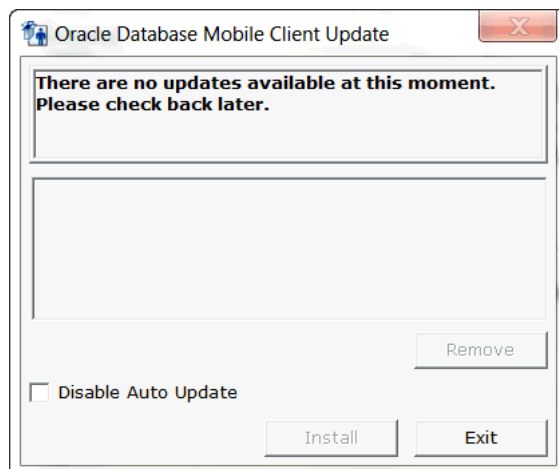
1. The administrator copies the patch INF files to the patch directory.
2. The administrator copies the new application files to the application directory.
3. The client user synchronizes with mobile server. Alternatively, the client user can invoke `update.exe` to check for the latest version of the software.

The mobile server checks for patches and sends all new patches to the client device.

7.6.3 Initiate Updates of Oracle Database Mobile Server Software from the Client

You can initiate a request for software updates from the mobile server by executing the Update tool, as shown in [Figure 7-8](#). To execute, choose "Update" from the Oracle Database Mobile Server Programs list or enter `update` on the command line.

The mSync tool automatically launches this tool if and only if a software update is available.

Figure 7–8 Updating Mobile Client 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 7.6.1.1, "Allowing Automatic Software Updates for the Oracle Database Mobile Server Platform"](#) for more information.

Note: In Windows Vista, Windows 7, and Windows 8 if the user has run update.exe with administrator privilege before, the dmagent.exe will also run as admin, then the next update with normal privilege will fail.

To run update.exe with normal privilege successfully, the user must stop dmagent before running update.exe.

Alternatively, you can initiate a request for software updates from the Mobile Server with the command line:

```
update /qs <Mobile Server URL> <user name> <password> [http proxy]
*/qs: checks available software updates from Mobile Server and installs all the updates quietly.
```

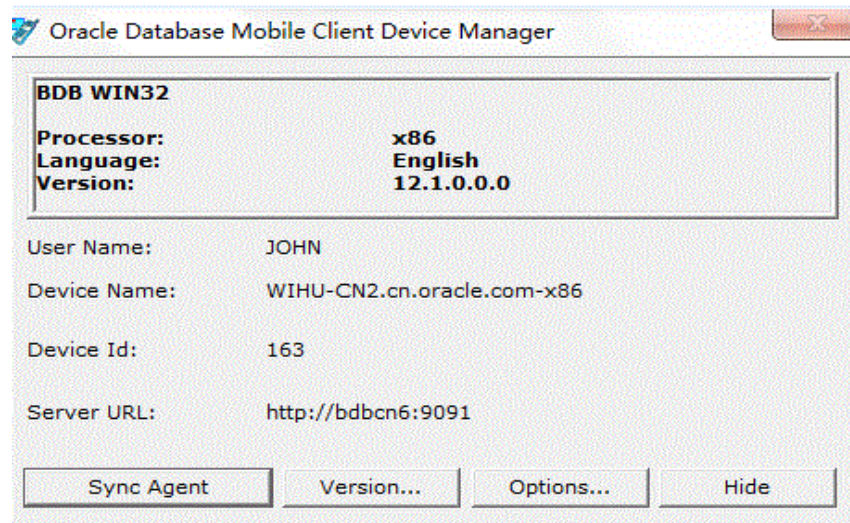
With this option, the software update prompt in [Figure 7–8](#) is skipped.

For example:

```
* update /qs http://mobile_server_host:7001/mobile JOHN welcome1
```

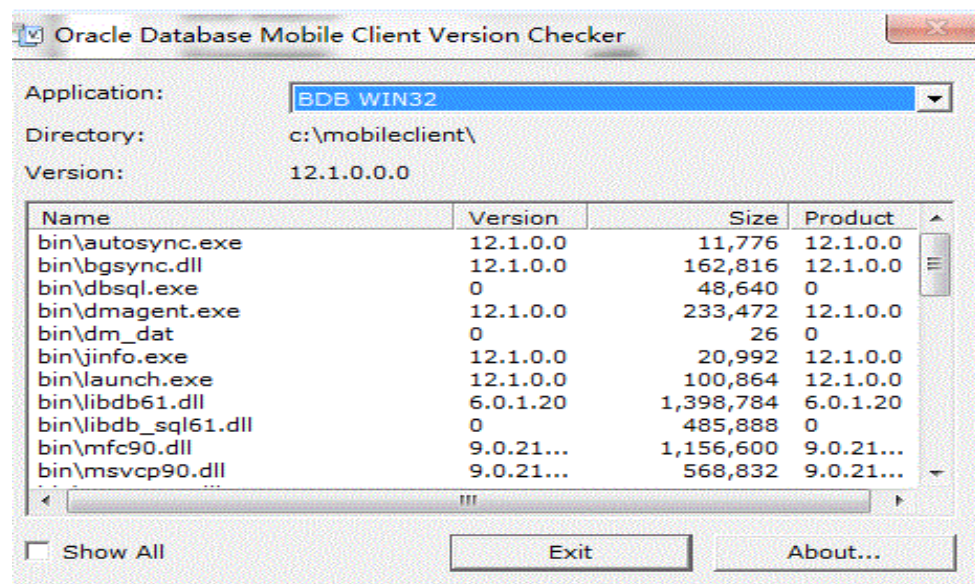
7.7 Using the Device Manager Agent (dmagent) on the Client

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 7–9](#).

Figure 7–9 Using the Device Manager to Manage Your Device

To bring up the Device Manager Agent GUI, choose "Oracle Database Mobile Server Device Manager" from the Oracle Database Mobile Server 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 "Sync Agent" to bring up the Sync Agent UI, which is described in [Section 5.5.2, "Start, Stop, or Get Status for Automatic Synchronization"](#).
- Click "Version" to see the version number of all mobile client software executables and DLLs, as shown in [Figure 7–10](#).

Figure 7–10 Mobile Client Software Versions

- 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 7.6.3, "Initiate Updates of Oracle Database Mobile Server 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

The screenshot shows a dialog box titled "Options" with three main sections:

- General:**
 - Enable Pull
 - Frequency: 1800
 - Retry Count: 3
- Check for Update:**
 - Never @ 00:00
- Proxy Server:**
 - Name: []
 - Port: []

Buttons: Save, Exit

- Click "Hide" to place the Oracle Database Mobile Server Device Manager in the Windows System Tray.

7.8 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 Windows Mobile, 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 connection to the mobile server.

- RAPI—Remote API used by the ActiveSync API, which only supports Windows Mobile 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 4.5 or Windows Mobile Device Center 6.1 depending on the Windows Operating System version.
 - Ensure CEUTIL.DLL and RAPI.DLL are in the %WINDIR%\System32 directory.
- SMS—Short Messaging Service.

Android client only supports SMS as the network provider.

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 9.8.1.5.3, "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 `NetworkProvider` 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".

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

Note: The server-side INF files that you can modify for the client platform are located in the <MOBILE_HOME>\Mobile\Server\admin\repository\setup\dmc directory.

As [Table 7–2](#) describes, the supported Software Management Client (SMC) 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_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, which can be one of the following: X86, ARMV4 or ARMV4I.
\$HOST_NAME\$	Host name of the client device.
\$USER_NAME\$	Mobile Server User Name
\$HTTP_PROXY\$	HTTP Proxy Server URL, if any.
\$SERVER_URL\$	mobile server URL.
\$USER_HOME\$	Home directory of the login user, which is determined by the value of %UserProfile% and \$HOME environment variable on Windows and Linux, respectively.

The following sections describe the INF file:

- [Section 7.9.1, "Setup Information"](#)
- [Section 7.9.2, "Properties"](#)
- [Section 7.9.3, "Initialization"](#)
- [Section 7.9.4, "Including Other INF Files"](#)

- [Section 7.9.5, "INSTALL Element"](#)

7.9.1 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="SQLite" 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.

7.9.2 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>
```
- `USERS`—Optional setting that defines under what privileges to install the mobile client. Also, if not configured in the INF file, then a prompt will appear in the installation to ask under what privileges to install the client.

The `prompt` attribute describes if the screen prompt for asking under what privileges to install the mobile client should be displayed. The `prompt` attribute defaults to `true`. Setting the `prompt` attribute to `false` eliminates this screen from displaying during client installation.

The two options for user install privilege are as follows:

- **Install for all users:** This requires an administrator to install as it provides access to the main user on the device. The following is an example of how to set this privilege:

```
<users prompt='false'>All</users>
```

- **Install for single user:** This requires only the user privilege as only a single user is using the application and device. The following is an example of how to set this privilege:

```
<users prompt='false'>Current</users>
```

Note: If any other word is provided for the <USERS> setting, then it will default to `prompt='true'` and the user will be prompted for the user privilege.

The presence of the <users> section will not have any effect if the user is installed on Windows Mobile or Linux installation, which automatically requires the administration privilege.

- 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 the 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 platforms 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 `sqlite3.dll` when you are installing the client, the prompt "Would you like to terminate the application?" is provided to the user.


```
<prompt><item type='WINCE' file='sqlite3.dll' />Would you like to terminate
the application?</prompt>
```

The following is an example of the setup section in the INF file:

```
<setup name="SQLite" version="1.0.0.0">
<property>
  <storage>4</storage>
  <memory>12</memory>
  <location>d:\tmp\a</location>
  <users prompt='false'>Current</users>
  <prompt>
    <item>Would you like to install App1?</item>
    <item file='sqlite3.dll'>Would you like to close the
      applications?</item>
  </prompt>
</property>
```

7.9.3 Initialization

Initialization includes setting keywords that you can use when installing your application; the Oracle Database Mobile Server 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>
```

7.9.4 Including Other INF Files

The following syntax allows an INF file to include other INF files:

```
<include>/dmc/common/devmgr.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.

7.9.5 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>
```

Where `msg_i` is the message to be shown during install when an action is executed and `msg_u` is the message to be shown during uninstall when an action is executed.

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	JDK 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.

7.9.5.1 DIRECTORY Section

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>$APP_DIR$\sqlite_db</item>
  <item>$APP_DIR$\crm</item>
</directory>
```

The SMC creates `sqlite_db` and `CRM` directories in the `ROOT` of the application directory.

7.9.5.2 FILE Section

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>

```

7.9.5.3 ENV Section

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'>$APP_DIR$\MYAPP</item>
</env >

```

The above example appends the `application_root\myapp` directory to the `PATH` environment variable.

7.9.5.4 REGISTRY Section

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.

7.9.5.5 ODBC Section

This section creates/registers the ODBC driver and DSNs for the mobile client device.

The following example registers and ODBC driver for the Berkeley DB Mobile Client:

```

<odbc>
  <item name="driver:BDB ODBC Driver" dll='$APP_DIR$\bin\sqlite3odbc.dll'>
    <version>00.86</version>
    <admin>$APP_DIR$\bin\sqlite3odbc.dll</admin>
  </item>
  <item name="dsn:MY_APP" driver='BDB ODBC Driver'
    dll='$APP_DIR$\bin\sqlite3odbc.dll'>

```

```

    <Data_Directory>$APP_DIR$\..\bdb\data\USER_NAME$\my_app.db</Data_Directory>
  </item>
</odbc>

```

The following example registers and ODBC driver for the SQLite Mobile Client:

```

<odbc>
  <item name="driver:SQLite ODBC Driver" dll='$APP_DIR$\bin\sqlite3odbc.dll'>
    <version>00.86</version>
    <admin>$APP_DIR$\bin\sqlite3odbc.dll</admin>
  </item>
  <item name="dsn:MY_APP" driver='SQLite ODBC Driver'
    dll='$APP_DIR$\bin\sqlite3odbc.dll'>
    <Data_Directory>$APP_DIR$\..\sqlite\sqlite_db\USER_NAME$\my_app.db
    </Data_Directory>
  </item>
</odbc>

```

7.9.5.6 LINK Section

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 of libdb_sql.so on a UNIX platform in \$APP_DIR/bin directory, which points to \$APP_DIR\$/bin/libdb_sql61.so.

```

<link>
  <item name='libdb_sql.so'>
    <folder>$APP_DIR/bin</folder>
    <file>$APP_DIR$/bin/libdb_sql61.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 My application.lnk, where the .lnk is automatically appended, in the Startup folder for the MyApp.exe file, which is located in the \$APP_DIR/bin directory.

```

<link>
  <item name='My application'>
    <folder>Startup</folder>
    <file>$APP_DIR$\my_dir\MyApp.exe</file>
    <directory>$APP_DIR\bin</directory>
  </item>
</link>

```

7.9.5.7 INI Section

The INI 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, the following adds the RESUME parameter and modifies the DATA_DIRECTORY parameter in the mobile client OSE.INI file:

```

<ini>
  <item name="OSE.INI" section="OSE">
    <item name="RESUME">YES</item>
  </item>
  <item name='OSE.INI' section='SQLITE'>
    <item name="DATA_DIRECTORY">$APP_DIR$\sqlite</item>
  </item>
</ini>

```

```

    </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 `DataDirectory` 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 `DataDirectory` if it does not exist, then set the `replace` attribute to `false`. The following example only adds `DataDirectory` with `c:\mobileclient` if `DataDirectory` 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`.

7.9.5.8 EXECUTE Section

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 names, then the installer waits for the executable (in this case, the `MyApp.exe`) to post that event, before the installer moves on to the next action. For example,

```

<execute>
  <item>
    <file>$APP_DIR$\my_dir\MyApp.exe</file>
    <args>-h</args>
    <wait>MySetupExit/MySetupStop</wait>
  </item>
</execute>

```

7.9.5.9 REGISTER Section

The `REGISTER` element lists all DLLs to be registered with the Windows Operating System. For example,

```

<register >
  <item>$APP_DIR$\my_dir\MyApp.dll</item>
</register>

```

7.9.5.10 FINISH Section

The `FINISH` element controls what can happen during the last stage of processing an INF file. It controls whether or not to restart the computer after the client install, whether or not to show a completion message in the UI, or whether or not to open a specific URL in a browser.

The `FINISH` section must be added to the main platform `.inf` file, located in the `MOBILE_HOME\mobile\server\admin\repository\setup\dmc\` directory. The `FINISH` element contains two parameters, namely, `"restart"` and `"show"`. By default, `"restart"='false'` and `"show"='false'`. These can be set to `true` or `false` depending on what behavior is desired. The item contains the URL to open in a browser.

The following is a complete example of MOBILE_HOME\mobile\server\admin\repository\setup\dmc\bdb_win32.inf with the FINISH section added.

BDB Win32 client will show the installation completion message 'Completed software installation' in UI and open <http://localhost:80/mobile> in default browser but will not restart the computer.

For win32 client, the browser used is defined by the windows system.

For linux client, the browser used is /usr/bin/firefox. If firefox is not installed, then you need to install it.

For Windows Mobile client, opening a specific URL is not supported.

```
<setup name="BDB WIN32" version="12.1.0.0">
  <property>
    <storage>25</storage>
    <memory>20</memory>
    <location/>
    <prompt/>
    <patch>patches\bdb\win32</patch>
  </property>
  .
  <include>common/bdb_win32.inf</include>
  <install>
    <action msg_i='$FINISH_I$' msg_u='$FINISH_U$'>finish</action>
    <finish restart='false' show='true'>
      <item>http://localhost:80/mobile</item>
    </finish>
  </install>
</setup>
.
```

FINISH element is supported for Windows Mobile, Win32 and Linux mobile client, but is not supported for Android mobile client.

7.10 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 7.10.1, "Device Manager Tag Language Data Types"](#)
- [Section 7.10.2, "Operators That You Can Use With the Device Manager Tag Language"](#)
- [Section 7.10.3, "Syntax for the Device Manager Tag Language"](#)

- [Section 7.10.4, "Conditional Statements"](#)
- [Section 7.10.5, "Define Custom Functions"](#)
- [Section 7.10.6, "Manage the Database Connection"](#)
- [Section 7.10.7, "Global Classes"](#)
- [Section 7.10.8, "Importing Another OTL Page"](#)
- [Section 7.10.9, "Error Handling"](#)
- [Section 7.10.10, "Sample Device Manager Commands Using the Tag Language"](#)

7.10.1 Device Manager Tag Language Data Types

The allowed data types for the device manager are as follows:

7.10.1.1 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()`.

7.10.1.2 Number

The Number data type represents either an integer, a double (float), or a large number.

7.10.1.3 Integer

The Integer object represents a four byte signed value. It is a primitive data type with no public methods.

7.10.1.4 Long

The Long object represents an eight byte signed value. It is a primitive data type with no public methods.

7.10.1.5 Double

The Double object represents a signed double (float) value. It is a primitive data type with no public methods.

7.10.1.6 Boolean

The Boolean object has only two possible values of `true` or `false`. It is a primitive data type with no public methods.

7.10.1.7 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="${str.Tokenize (';')}">  
  Token = <c:out value="${tok}"/>  
</c:foreach>
```


7.10.1.8 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.

7.10.1.9 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.

7.10.1.10 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.

7.10.1.11 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.

7.10.1.12 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`.

7.10.2 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
+	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.

7.10.3 Syntax for the Device Manager Tag Language

OTL supports all regular scripting language syntax rules, such as Assignment, Conditional Constructs, and Sub Routines.

7.10.3.1 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" index="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 var="arr1" index="10" value="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')}"/>
```

7.10.3.2 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 01-02-2004.

7.10.3.2.1 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"/>
```

7.10.3.2.2 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}"/>
```

7.10.4 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)`.

7.10.4.1 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>
```

7.10.4.2 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>
```

7.10.4.3 Foreach Conditional Statement

The foreach conditional statement enables to you enumerate built-in enumeration objects, such as SQL Result Set and Vectors.

```
<c:foreach var="row" items="{rs}">
  <c:out value="{row[0]}" />
</c:foreach>
```

Also, this statement is used to execute a block of statements repeatedly by stepping through a STEP value.

```
<c:set var="count" value="100" />
<c:foreach var="row" begin="1" end="{count}" step="3">
  <c:out value="{row}" />
</c:foreach>
```

7.10.4.4 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:if>
</c:foreach>
```

7.10.4.5 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>
```

7.10.5 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 reflected 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}"/>
```

7.10.6 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.

7.10.6.1 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="MyDSN" "/>
```

7.10.6.2 Disconnect from the Database

To disconnect from the database, then issue the following:

```
<c:database action="disconnect"/>
```

7.10.7 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 7.10.7.1, "Methods of the System Class"](#): Use to access operating system information.
- [Section 7.10.7.2, "Methods of the DeviceManager Class"](#): Use to access device manager information.

7.10.7.1 Methods of the System Class

You can use the following system functions in your device manager command:

- [Section 7.10.7.1.1, "Retrieve HTTP Request Parameters and Session Values"](#)
- [Section 7.10.7.1.2, "Create a Date Object"](#)
- [Section 7.10.7.1.3, "Create a Time Object"](#)
- [Section 7.10.7.1.4, "Get, Set, or Remove Session Attributes"](#)
- [Section 7.10.7.1.5, "Retrieving Parameter Name or Value"](#)
- [Section 7.10.7.1.6, "Retrieving the Request URL"](#)
- [Section 7.10.7.1.7, "Retrieving the Last Error Message"](#)
- [Section 7.10.7.1.8, "Retrieving System Memory Information"](#)
- [Section 7.10.7.1.9, "Retrieving Storage Information"](#)
- [Section 7.10.7.1.10, "URL Encoding a String"](#)
- [Section 7.10.7.1.11, "Opening a File"](#)
- [Section 7.10.7.1.12, "Synchronizing Databases"](#)

7.10.7.1.1 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](#)
- [Retrieve HTTP Session Attributes](#)

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 `GetParameterNames` 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 `GetParameterNames` and `GetParameterValue`, see [Section 7.10.7.1.5, "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 7.10.7.1.4, "Get, Set, or Remove Session Attributes"](#).

7.10.7.1.2 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.

7.10.7.1.3 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.

7.10.7.1.4 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)`.

7.10.7.1.5 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.

7.10.7.1.6 Retrieving the Request URL Use the `System.GetURL()` method for retrieving the request URL.

7.10.7.1.7 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.

7.10.7.1.8 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.

7.10.7.1.9 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.

7.10.7.1.10 URL Encoding a String Encodes the provided string.

System.URLEncode (String value)

Returns the given string as a URL encoded string.

7.10.7.1.11 Opening a File Use `System.OpenFile (String name)` to open the file provided in the method parameter. Returns the `File` object.

7.10.7.1.12 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>
```

7.10.7.2 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 mobile client.

```
<c:set var='dir' value='\${DeviceManager.GetBinaryDir()}' />
```

DeviceManager.GetUserName ()

Returns the Oracle Database Mobile Server user name

```
<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 an OTL script of the name: sync.otl.

```
<c:set value='${DeviceManager.CreateCommand ("sync")}'/>
```

DeviceManager.GetRegistry (String key, String name)

Retrieve a value from the configuration file (OSE.INI). All the values are retrieved as String. The following example retrieves the value for DATA_DIRECTORY configured in the OSE.INI file.

```
<c:set var='val' value='${DeviceManager.GetRegistry ("SQLITE", "DATA_DIRECTORY")}'/>
```

DeviceManager.SetRegistry (String key, String name, String value)

Set a new configuration value in the configuration file (OSE.INI). The following example sets a new value for RESUME.

```
<c:set value='${DeviceManager.SetRegistry ("OSE", "RESUME", "YES")}'/>
```

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++, 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 OMCAPAPI.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);
}

```

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:

```

sqlplus <mobile server schema>/<password>
insert into dm$all_providers values ('MY_HANDLER', 'MESSAGE', 'MyMessage', NULL);

```

7.10.8 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>

```

7.10.9 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>

```

7.10.10 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", "Wednesday",
    "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 02-20-2002.

```
<c:set var="d" value="{date ('02-20-2002')}" />
```

7.11 Using Mobile Manager to Manage iOS Devices

iOS Mobile Device Management (MDM) functionality is supported for iOS 7 devices. To manage an iOS device from mobile manager, the device must first be enrolled. After it is enrolled, mobile manager administrator can view the device information, send commands to the device, and view the command history from history list.

This section describes the following:

- How to enroll iOS device, see, [Section 7.11.1, "iOS Device Enrollment"](#)
- How to view iOS device information on mobile manager, see, [Section 7.11.2, "View iOS Device Information"](#)
- iOS MDM command management, see, [Section 7.11.3, "iOS MDM Command Management"](#)
- iOS device profile management, see, [Section 7.11.4, "iOS Profile Management on Mobile Manager"](#)

Note: Device enrollment is available for all iOS device users but the other functionalities are available only for mobile manager administrators.

7.11.1 iOS Device Enrollment

To enroll an iOS device in MDM server, follow the steps below on the iOS device:

1. Go to the setup page:

https://server_ip:server_https_port/mobile/console/setup/setuppage.uix

- Choose Language and Platform for the device type you need to enroll as shown in [Figure 7-12](#). Click "BDB iOS" or "SQLite iOS" to start your enrollment. It will go to iOS MDM enrollment page:

https://server_ip:server_https_port/mobile/console/setup/MdmRegisterLogin.uix

See [Figure 7-13](#).

You can access the iOS MDM enrollment page by clicking "BDB iOS" or "SQLite iOS" on the setup page. You cannot directly access the page.

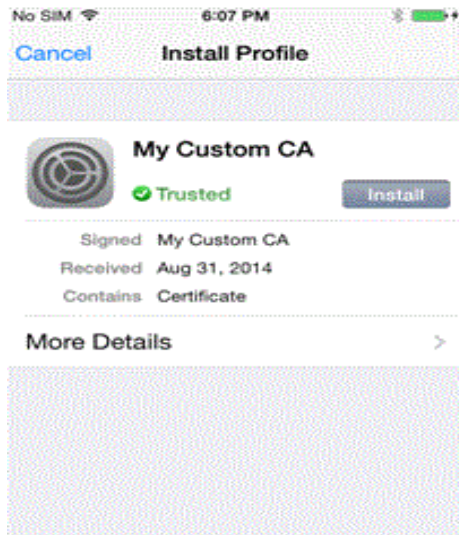
Figure 7-12 Setup Page

Mobile Client	Language
BDB iOS Sync	English
BDB iOS	English
SQLite iOS Sync	English
SQLite iOS	English

Figure 7-13 iOS MDM Enrollment Page

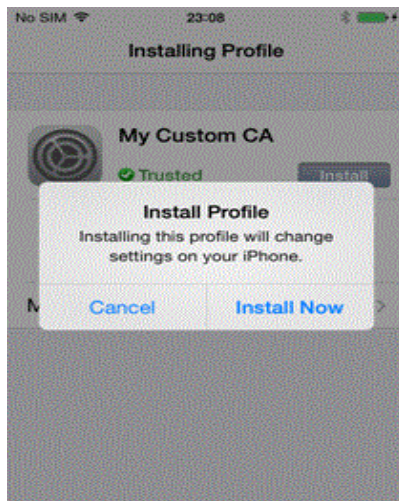
- Click "Get CA Certificate" on the iOS MDM enrollment page, to get Certificate Authority (CA) certificate downloaded on the iOS device. The following steps describe the procedure on the device.
 - Click "Install" as shown in [Figure 7-14](#).

Figure 7–14 Install CA Certificate - Step 1

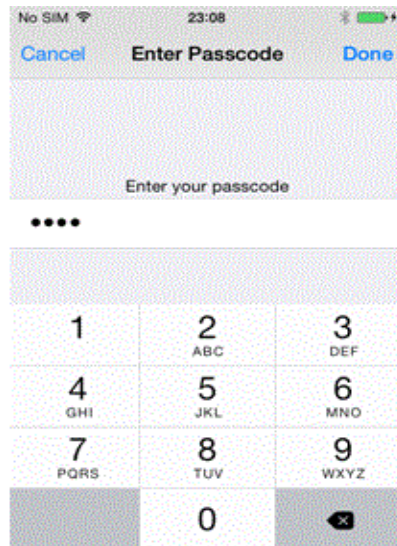


- Click "Install Now" as shown in [Figure 7–15](#).

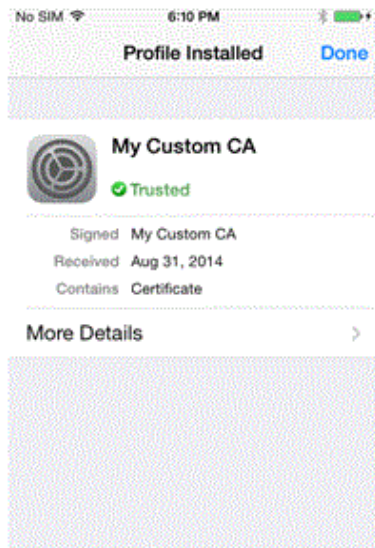
Figure 7–15 Install CA Certificate - Step 2



- Enter "Passcode" as shown in [Figure 7–16](#).

Figure 7–16 Install CA Certificate - Step 3

- Click "Done" as shown in [Figure 7–17](#). The CA certificate is downloaded and installed successfully.

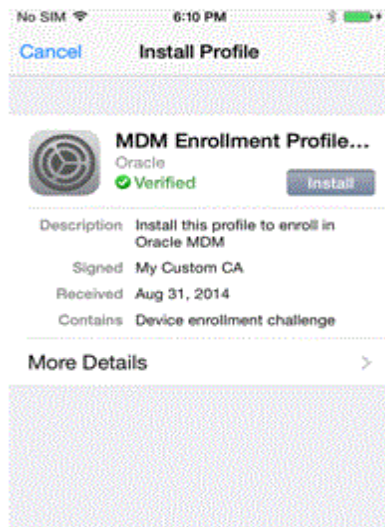
Figure 7–17 Install CA Certificate - Step 4

- Enter the username and password on iOS MDM enrollment page (after the CA certificate is installed) as show in [Figure 7–13](#) and click "Enroll".

The enrollment process will install all published configuration and provisioning profiles with MDM payload and will also install all applications granted to this user for the given platform. The following steps illustrate this:

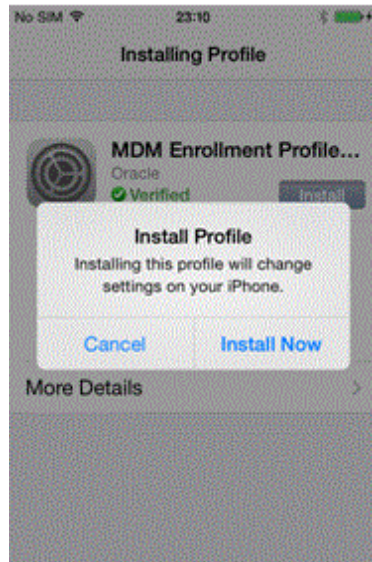
- Click button "Install" as shown in [Figure 7–18](#).

Figure 7–18 Enroll - Step 1

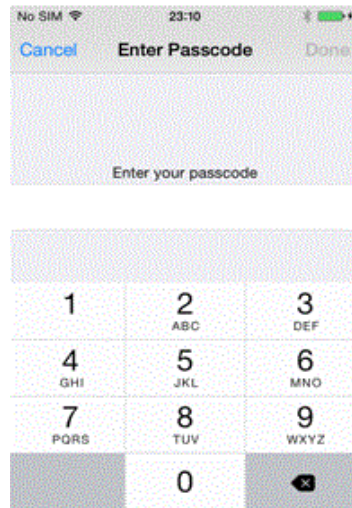


- Click "Install Now" as shown in [Figure 7–19](#).

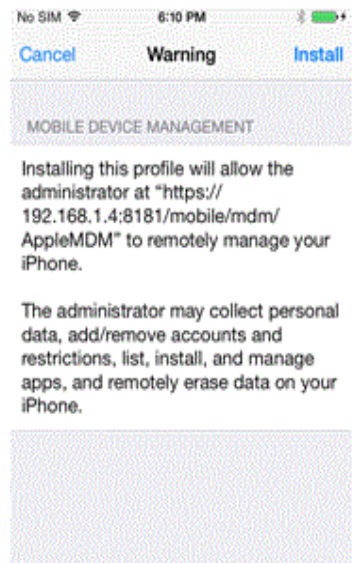
Figure 7–19 Enroll - Step 2



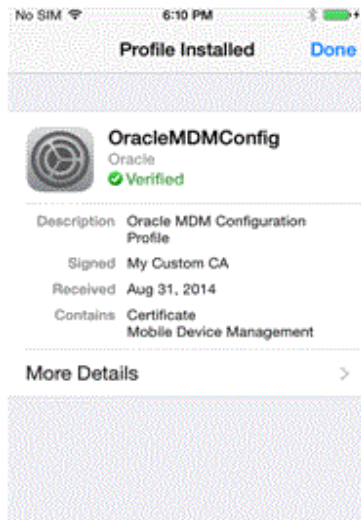
- "Enter Passcode" as shown in [Figure 7–20](#).

Figure 7-20 Enroll - Step 3

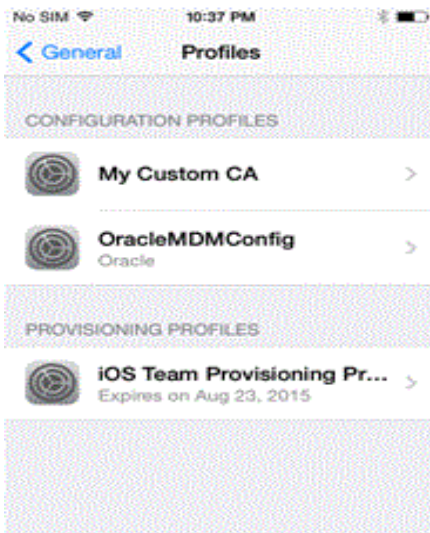
- Click "Install" as shown in [Figure 7-21](#).

Figure 7-21 Enroll - Step 4

- Click "Done" as shown in [Figure 7-22](#). The configuration profile is installed and the enrollment is done successfully.

Figure 7–22 Enroll - Step 5

After the enrollment is finished on the device, all profiles that are installed during enrollment can be found on the device in iOS settings "General" Profiles as shown in [Figure 7–23](#). The device is listed in Devices page on the mobile manager.

Figure 7–23 The Profiles Installed During Enrollment

7.11.2 View iOS Device Information

Once an iOS device is enrolled, see [Section 7.11.1, "iOS Device Enrollment"](#), mobile manager administrators can view the device information on mobile manager.

Go to tab "Device Info" for that device as shown in [Figure 7–24](#). To refresh device and security information with update-to-date values, use the "Retrieve Device Information" button.

Figure 7–24 Device Info Page

Device: iPad

Properties Device Info Database Info Software Info Commands Queue Command History Device Logs

Page Refreshed Sep 10, 2014 10:15:49 PM

Information retrieved on Aug 26, 2014 6:57:09 AM

Retrieve device information

General Information

Name	Value
Device Name	iPad
Model Name	iPad
Model Number	MDS543LL
Serial Number	F4KK2AZMF19M
Build Version	11D201
Battery Level	80
Version	7.1.1
Total Size (KB)	13,020,295
Free Space (KB)	12,494,254
Unique Device ID	1ab148226040e129f79a7ee2f6d656622c8a6570

Network

Name	Value
Carrier Network	Verizon
Sim Card ID	8914 8000 0004 2464 1810
Phone Number	4085980706

Security

Name	Value
Unique Device ID	1ab148226040e129f79a7ee2f6d656622c8a6570
Passcode Present	false
Passcode Compliant	true

You can view application information and status on tab "Software Info" as shown in Figure 7–25. To retrieve update-to-date information for all applications on the device, use the "Retrieve Software Information" button.

Figure 7–25 Software Info Page

Properties Device Info Database Info Software Info Commands Queue Command History Device Logs

Page Refreshed Sep 16, 2014 11:21:45 PM

Retrieve software information

Information retrieved on Sep 16, 2014 11:21:45 PM

Name	Bundle ID	Version	Bundle Size	Application Data Size	Status
TransportDemo	com.oracle.TransportDemo	2.1	2867200	90112	Managed

Properties Device Info Database Info Software Info Commands Queue Command History Device Logs

7.11.3 iOS MDM Command Management

iOS MDM command management is available for mobile manager administrator. For iOS devices, the following command management is supported:

- Section 7.11.3.1, "All Commands"
- Section 7.11.3.2, "Send Command from Mobile Manager"
- Section 7.11.3.3, "View Command History"
- Section 7.11.3.4, "Differences between iOS Device Management and Non-iOS Device Management"

7.11.3.1 All Commands

You can retrieve commands from tab *Mobile Devices -> Administration -> Command Management*.

7.11.3.2 Send Command from Mobile Manager

The following commands can be sent from mobile manager:

- **Device Information**

Retrieve device information from iOS device. Once the command is executed, the device information can be retrieved on the device information page as shown in [Figure 7-24](#).
- **Retrieve iOS Security Information**

Retrieve iOS security information. Once the command is executed, the security information can be viewed on the device information page as shown in [Figure 7-24](#).
- **Install Application**

Install an application on iOS device. The application must be published correctly to mobile server, its platform must match the device platform and the user must have access to the application.
- **Retrieve iOS Application Status**

Retrieve application status for applications which are installed on the device. It can retrieve status on a specific application or retrieve status on all applications. Once the command is executed, the application status can be viewed on software information page as shown in the "Status" column of [Figure 7-25](#).
- **Retrieve Software Information**

Retrieve application information for applications which are installed on the device. Once the command is executed, the application status can be viewed on software information page as shown in all other columns of [Figure 7-25](#).
- **Remove iOS Application**

Remove an application which is installed on the device.
- **Update iOS Application**

Update an application which is installed on the device.
- **Update all iOS Applications**

Update all applications which are installed on the device.
- **Install Config Profile for iOS Device**

Install a configuration profile on the device. The profile needs to be uploaded in the repository before the command is sent to devices. See [Section 7.11.4.1, "Create or Update Profiles,"](#) for information on how to create or update profiles.
- **Install Provisioning Profile for iOS device**

Install a provisioning profile on the device. The profile needs to be uploaded in the repository before the command is sent to devices. See [Section 7.11.4.1, "Create or Update Profiles,"](#) for information on how to create or update profiles.
- **Remove Config Profile for iOS Device**

Remove a configuration profile from the device.
- **Remove a Provisioning Profile for iOS Device**

Remove a provisioning profile from the device.
- **Lock iOS Device**

Lock the device. If a message is entered when the command is sent, the message will be displayed on the device when the device is locked. Otherwise, the default message "Device Locked by MDM" will be displayed when the device is locked.

- Erase iOS Device
- Clear Passcode for iOS device.
- Unenroll iOS Device from MDM

Unenroll the device from mobile server. Once the command is executed, the record of this device is deleted from the mobile repository.

Commands listed above can be viewed in the drop down list in tab "Commands". The following are examples for sending MDM commands from the mobile manager.

Example 1: Send "Install Config Profile for iOS Device"

- Go to the tab "Commands" for the device that will receive the command. Choose "Install Config Profile for iOS Device" as shown in [Figure 7–26](#). Click "Send Now".

Figure 7–26 Install Configuration Profile - Step 1

The screenshot shows the 'Commands' tab in the Mobile Manager interface. At the top, there are navigation tabs: Properties, Device Info, Database Info, Software Info, Commands (selected), Queue, Command History, and Device Logs. Below the tabs, it says 'Page Refreshed Aug 29, 2014 12:13:06 AM' and a 'Schedule' button. A table with columns 'Select Job Name', 'Repeat Mode', 'Start', 'Execution Count', 'Command', and 'Status' is shown, with '(No items found)' in the first column. Below the table is a 'Send Command' section with a dropdown menu set to 'Install Config Profile for iOS Device' and a 'Send Now' button. At the bottom, the same navigation tabs are visible.

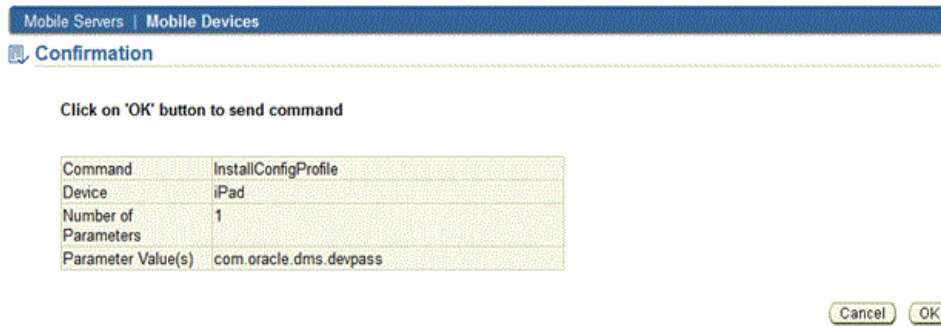
- All available configuration file names in the mobile repository are listed in the drop down list. Choose the one to be installed and click "Next" as shown in [Figure 7–27](#). If no configuration files are available in the repository, refer to [Section 7.11.3.1, "All Commands"](#) to upload a new profile.

Figure 7–27 Install Configuration Profile - Step 2

The screenshot shows the 'Mobile Servers | Mobile Devices' section. Below it is an 'Information' icon and the text 'Provide the following parameter(s):'. A 'Profile Name' dropdown menu is shown, with 'Device passcode profile(com.oracle.dms.devpass)' selected. At the bottom right, there are 'Cancel' and 'Next' buttons.

- The command and parameter will be displayed for confirmation as shown in [Figure 7–28](#). Click "OK" to send the command. The command is then sent to the device.

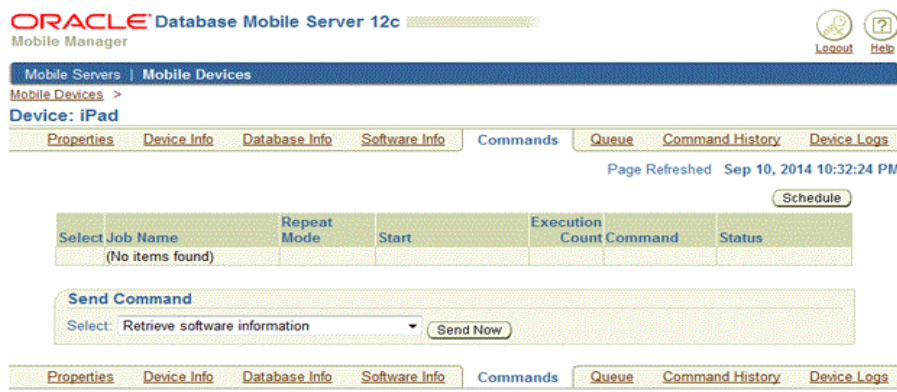
Figure 7–28 Install Configuration Profile - Step 3



Example 2: Send "Retrieve Software Information"

- Go to the tab "Commands" for the device that will receive the command. Choose "Retrieve Software Information" as shown in [Figure 7–29](#). Click "Send Now".

Figure 7–29 Retrieve Software Information - Step 1



- The drop down list shows all application names that the users have access to and the application's platform matches the device's platform. If no such application exists in the repository, a warning message will be displayed. Applications must be published using packaging wizard and is assigned to the user.

Choose the application name that you want to retrieve information from the drop down list to retrieve the specific application information. You can also choose "All Applications" to retrieve all applications that are installed on the device as shown in [Figure 7–30](#). Click "Next" to continue.

Figure 7–30 Retrieve Software Information - Step 2

Information

Provide the following parameter(s):

Application Name

Select: All Applications

Cancel Next

- Click "Yes" to send the command as shown in [Figure 7–31](#).

Figure 7–31 Retrieve Software Information - Step 3

Mobile Servers | Mobile Devices

Confirmation

Click on 'Yes' button to send command SoftwareInfo

No Yes

Once the command is executed successfully, updated application information can be retrieved as described in [Section 7.11.2, "View iOS Device Information"](#).

7.11.3.3 View Command History

Click tab "Command History". The page lists all commands which have been sent to the device as shown in [Figure 7–32](#).

Figure 7–32 Command History

Mobile Servers | Mobile Devices

Mobile Devices >

Device: iPad

Properties Device Info Database Info Software Info Commands Queue Command History Device Logs

Page Refreshed Sep 10, 2014 10:34:48 PM

Send Command Purge Delete

Select All | Select None

Select	Command	Send Time	Status Message	Status Time	Verification Time
<input type="checkbox"/>	Install Provisioning Profile for iOS Device		Push Notification Sent	Aug 26, 2014 6:57:09 AM	
<input checked="" type="checkbox"/>	Retrieve iOS Security Information		Command Succeeded	Aug 26, 2014 6:57:09 AM	
<input checked="" type="checkbox"/>	Device information		Command Succeeded	Aug 26, 2014 6:57:08 AM	
<input type="checkbox"/>	MDM bootstrap command		Command Execution Started	Aug 26, 2014 6:57:07 AM	

Properties Device Info Database Info Software Info Commands Queue Command History Device Logs

7.11.3.4 Differences between iOS Device Management and Non-iOS Device Management

See the sections below:

- [Section 7.11.3.4.1, "Device Management on Devices"](#)
- [Section 7.11.3.4.2, "Device Management on Mobile Manager"](#)

7.11.3.4.1 Device Management on Devices A device manager agent is not needed for iOS devices. For non-iOS devices, a device manager agent (dmagent) runs on native clients.

7.11.3.4.2 Device Management on Mobile Manager The following functionalities on mobile manager are not supported for iOS devices:

- [Section 7.1, "Customize the Mobile Client Software Installation for Your Mobile Device"](#)
- [Section 7.2, "Configuring Mobile Clients Before Installation"](#)
- [Section 7.3.3, "Viewing Database Information"](#)
- [Section 7.3.6, "Queue"](#)
- [Section 7.4, "Configuring and Customizing Your Mobile Device Platform"](#)
- [Section 7.5.1.1.1, "Description of Existing Commands"](#). However, the commands described in [Section 7.11.3.2, "Send Command from Mobile Manager"](#) are supported for iOS devices.
- [Section 7.5.2, "Modifying Existing Commands"](#)
- [Section 7.5.3, "Creating New Commands"](#)
- [Section 7.6.1.2, "Updating Application Software On Each Client"](#)
- [Section 7.6.2, "Configuring Application Software for Automatic Update"](#)
- [Section 7.6.3, "Initiate Updates of Oracle Database Mobile Server Software from the Client"](#)
- [Section 7.7, "Using the Device Manager Agent \(dmagent\) on the Client"](#)
- [Section 7.8, "Managing the Network Protocol Between the Device and the Mobile Client Software"](#)
- [Section 7.9, "Installation Configuration \(INF\) File"](#)
- [Section 7.10, "Defining Device Manager Commands With the Device Manager OTL Tag Language"](#)
- View device logs in Tab "Device Logs".

7.11.4 iOS Profile Management on Mobile Manager

You can manage iOS profiles in mobile manager by going to tab *Mobile Devices* -> *Administration* -> *iOS Device Profile Management*. iOS profile management includes creating or updating profiles in mobile repository, deleting profiles from mobile repository, and retrieving profile information. The functionalities are supported for both configuration profiles and provisioning profiles.

The configuration profile management page is:

https://server_ip:server_https_port/mobile/console/DMAAdmin-Profile.uix

Provisioning profile management page is:

https://server_ip:server_https_port/mobile/console/DMAAdmin-ProvisProfile.uix

The following sections describe the iOS profile management functionalities:

- [Section 7.11.4.1, "Create or Update Profiles"](#)
- [Section 7.11.4.2, "Delete Profiles"](#)
- [Section 7.11.4.3, "Retrieve Profile Information"](#)

7.11.4.1 Create or Update Profiles

To create or update a configuration/provisioning profile, follow the steps below on configuration/provisioning profile management page:

- Click "Create or Update Profile" on the page as shown in [Figure 7–33](#).
- Click "Browse" to retrieve a configuration/provisioning profile on the local computer for uploading.
- Click button "OK" to upload it.

If the configuration profile does not exist in the mobile repository, it creates a new profile in the repository according to the uploaded profile and the new profile is listed in configuration profile page.

If the configuration profile exists in the mobile repository, it displays a warning message "Do you want to overwrite the profile" as shown in [Figure 7–34](#). Click "OK" to overwrite the existing profile in the repository with the uploaded profile. Otherwise, no change is done in the repository.

Mobile manager checks the configuration profile id to decide whether a configuration profile exists in the repository or not while it checks the provisioning profile uuid to decide whether a provisioning profile exists in the repository or not.

The profiles are validated during uploading and invalid profiles are not be allowed to be stored in the repository. Configuration profiles are signed before getting stored in the repository. Provisioning profiles are downloaded from Apple Development Portal and should already be signed by Apple.

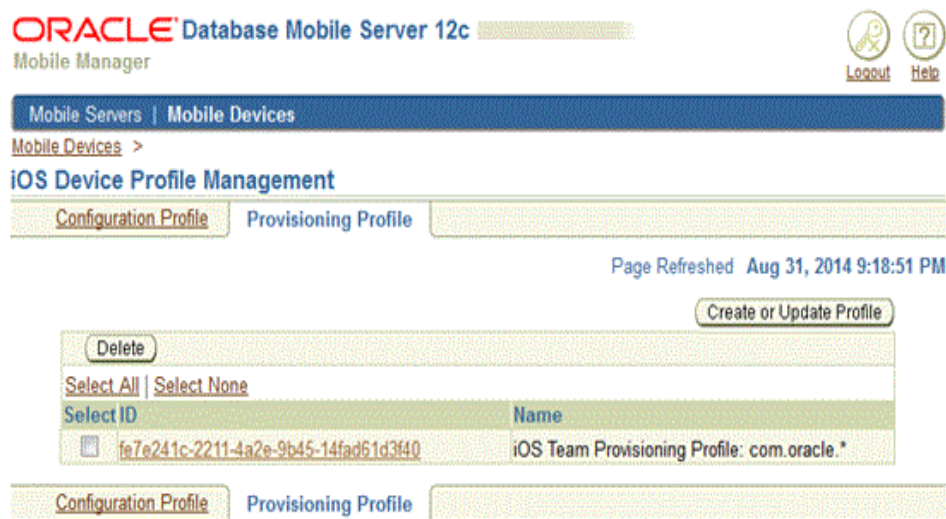
Figure 7–33 Configuration Profile Management Page



Figure 7–34 Warning Message for Overwriting an Existing Profile



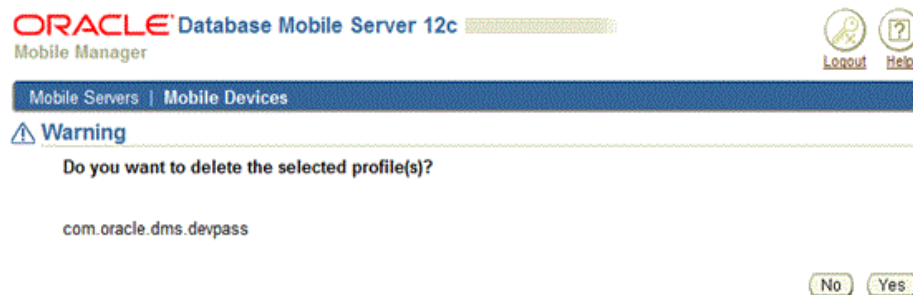
Figure 7–35 Provisioning Profile Management Page



7.11.4.2 Delete Profiles

To delete configuration/provisioning profiles from the mobile repository, select the profiles on configuration/provisioning profile management page and click button "Delete". A warning message "Do you want to delete the selected profile(s)" with profile id or profile UUID will be displayed as shown in Figure 7–36. If "Yes" is clicked, the selected profiles will be deleted from the mobile repository. Otherwise, no change is done in the repository.

Figure 7–36 Warning Message for Deleting Profiles



7.11.4.3 Retrieve Profile Information

To retrieve profile information, go to Configuration/Provisioning Profile Management page, and click the "Profile ID" in the Profile Table as shown in [Figure 7-37](#).

Figure 7-37 Profile Information

The screenshot displays the Oracle Database Mobile Server 12c Mobile Manager interface. At the top, the Oracle logo and "Database Mobile Server 12c" are visible, along with "Mobile Manager" and navigation icons for "Logout" and "Help". A breadcrumb trail shows "Mobile Servers" > "Mobile Devices" > "Mobile Devices" > "iOS Device Profile Management". The page is timestamped "Page Refreshed Aug 31, 2014 9:30:14 PM". The main content area is titled "Configuration Profile" and contains a table with the following data:

Name	Value
Configuration Profile ID	com.oracle.dms.devpass
UUID	D34F4145-4FE8-4A4B-A45C-5857D54EAB2D
Name	Device passcode profile
Description	Device passcode policy.
Organization	Oracle
Version	1

Offline Instantiation for Mobile Clients

The Offline Instantiation (OLI) utility enables mobile administrators to prepare a batch package that includes the mobile client software and initial data for every mobile user. The OLI package can be used to set up a mobile client with user-specific initial data. This procedure helps users avoid an expensive initial synchronization download to the mobile client.

The following sections discuss the Offline Instantiation feature.

- [Section 8.1, "Using Offline Instantiation to Distribute Multiple Mobile Clients"](#)
- [Section 8.2, "Setting Up the Mobile Server Host and Mobile Development Kit Host"](#)
- [Section 8.3, "Downloading the Mobile Client SETUP Executable"](#)
- [Section 8.4, "Configure How OLI Creates the Client Distribution Packages With the OLI Configuration File"](#)
- [Section 8.5, "Using the OLI Engine to Create and Package the Client Distribution File"](#)
- [Section 8.6, "Deploying Client Distribution Files on Client Machines"](#)

8.1 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 client binaries or the mobile client binaries with the user database files, including the user data. 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 and is only supported on a Windows platform. 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. **Create application:** 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. **Download the `setup.exe` file:** On the Windows machine where the Mobile Development Kit is installed, download the mobile client binary (`setup.exe`) from Mobile Manager. Choose the Windows platform and language that are appropriate for the mobile clients that you are creating.
 3. **Configure the `oli.ini` file:** Configure the `oli.ini` file to tell the offline instantiation batch tool how to create the client distribution packages. See [Section 8.4, "Configure How OLI Creates the Client Distribution Packages With the OLI Configuration File"](#) for directions.
 4. **Execute the OLI utility:** Use the offline instantiation tool (OLI) to create and package the final client distribution packages for each user. See [Section 8.5, "Using the OLI Engine to Create and Package the Client Distribution File"](#) for directions.
 5. **Distribute the package:** Distribute the client distribution packages to each user to install on their client device. See [Section 8.6, "Deploying Client Distribution Files on Client Machines"](#) for directions.

8.2 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:

Note: Do not install a mobile client on the same machine as the Mobile Development Kit where OLI will be executed. If a mobile client is present, uninstall it before running OLI.

1. Install the mobile server and Mobile Development Kit.

The disk where Mobile Development Kit is installed must have sufficient free space as this is the staging area where all client binaries and database files are created before they are copied to the final package. The free space should exceed the total size of the data to be packaged for all clients combined plus 200 MB. For example, if you want to package 50 clients where each uses 20 MB of data, then you need at least 1.2 GB of free space (50 x 20 MB + 200 MB = 1.2 GB).
2. Start the mobile server.
3. Create the appropriate users, publications, and subscriptions on the mobile server. Subsequent operations are carried out on the MDK host.

8.3 Downloading the Mobile Client SETUP Executable

On the Windows machine where the Mobile Development Kit is installed, download `setup.exe` into an empty directory—such as `C:\olisetup`—from the mobile server setup page. The `setup.exe` can be downloaded from the following URL:

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

Install the mobile client by running the downloaded setup.exe. The full path where you install the mobile client must be specified in the `SETUP_PATH` parameter in the `OLI.INI` file. Refer to [Section 8.4.1, "SETUP"](#) for more information on this parameter.

8.4 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\`, it includes a sample OLI configuration file named `oli.ini` and the OLI batch script named `oli.bat`.

The following describes how to configure the `OLI.INI` file:

- [Section 8.4.1, "SETUP"](#)
- [Section 8.4.2, "USERS"](#)
- [Section 8.4.3, "Example of OLI.INI File"](#)

8.4.1 SETUP

The `SETUP` section contains the general configuration for OLI to create the client packages.

SETUP_PATH

Define the absolute location where you downloaded the client `setup.exe`, as described in [Section 8.3, "Downloading the Mobile Client SETUP Executable"](#).

```
SETUP_PATH=C:\olisetup\setup.exe
```

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 Oracle database or Oracle RAC database where the mobile server repository exists.

For example, for Oracle Database 10g or 11g whose host, port and SID are `<host>`, `<port>` and `<SID>`:

```
jdbc:oracle:thin:@<host>:<port>:<SID>
```

For example, for Oracle Database 12c whose host, port and service name are `<host>`, `<port>` and `<ServiceName>`:

```
jdbc:oracle:thin:@//<host>:<port>/<ServiceName>
```

If the repository exists in an Oracle RAC database, then the JDBC URL for an Oracle RAC database can have more than one address in it for multiple Oracle databases in the cluster and follows this URL structure:

```
jdbc:oracle:thin:@(DESCRIPTION=
  (ADDRESS_LIST=
    (ADDRESS= (PROTOCOL=TCP) (HOST=PRIMARY_NODE_HOSTNAME) (PORT=1521) )
    (ADDRESS= (PROTOCOL=TCP) (HOST=SECONDARY_NODE_HOSTNAME) (PORT=1521) )
  )
  (CONNECT_DATA= (SERVICE_NAME=DATABASE_SERVICENAME) ) )
```

SCHEMA and PASSWORD

Configure the mobile server administration schema name and password for the mobile server repository.

```
SCHEMA=MOBILEADMIN_SCHEMA
PASSWORD=MOBILEADMIN_PASSWORD
```

OLI_CDS_ROOT

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. Configure this directory in the OLI_CDS_ROOT directory.

From this final directory—where OLI places all of the client distribution packages for each user—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. Provide device platform strings. Platform types are created by identifying the language and the mobile client database as follows:

Note: Currently, only Windows platforms are supported.

```
WIN32_X86_<LANGUAGE>_<DB>
```

The components of each of section for the device can be as follows:

Table 8–1 Device type platform components

Component	Options
<LANGUAGE>	JA, ES, KO, DE, US, FR, IT, ZHS, PTB
<DB>	Berkeley DB, SQLite

For example, a US Berkeley DB Mobile Client device type is as follows:

```
DEVICE_TYPE=WIN32_X86_US_BDB
```


8.4.2 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 8.1, "Using Offline Instantiation to Distribute Multiple Mobile Clients"](#). On each line, put the user name and password, as follows:

```
[USERS]

CONSC1 MANAGER
CONSC2 MANAGER
```

For each user, a client distribution package is created.

8.4.3 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 Database Mobile Server Offline Instantiation Configuration File
# Copyright © 1997-2011 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]

# Absolute path of setup downloaded from mobile server
#
SETUP_PATH=C:\olisetup\setup.exe

#
# 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:orcl
SCHEMA=MOBILEADMIN_SCHEMA
```

```

PASSWORD=MOBILEADMIN_PASSWORD

# Each user has its own package created under <OLI_CDS_ROOT>/<USERNAME>.
#
OLI_CDS_ROOT=C:\OLI_CDS

#
# The device type of the targeted mobile client machines.
#
DEVICE_TYPE=WIN32_X86_US_BDB

#
# 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

```

8.5 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 8.4, "Configure How OLI Creates the Client Distribution Packages With the OLI Configuration File"](#). The OLI engine uses two repository tables—`C$OLI_CLIENTS` and `C$OLI_SETUP`—that store information related to resuming OLI tasks during interruptions or failures.

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.

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.

This action displays the following usage information.

Note: You execute only ONE of the following commands at a time: `makedb`, `package`, `cleanup`, or `checkstatus`. 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 [-g] [makedb] [package] [cleanup] [checkstatus]

```

The `-g` command option for `oli.bat` turns on debugging.

Note: Before executing the `makedb` or `package` command on Windows Mobile or Win32 devices, ensure that you set the `DEVICE_TYPE` parameter in the `oli.ini` file.

To carry out OLI tasks, re-execute the command using the appropriate switches and arguments.

The following sections describe how to build the client installation package:

- [Section 8.5.1, "Create and Populate Client Database Files with the MAKEDB Command"](#)
- [Section 8.5.2, "Package the Mobile Client Binaries with the Client Database Files with the PACKAGE Command"](#)
- [Section 8.5.3, "Clean Up the OLI Tables Before Executing OLI for Another Distribution"](#)
- [Section 8.5.4, "Check the Status of OLI Clients"](#)

8.5.1 Create and Populate Client Database Files with the MAKEDB Command

Creates and populates the client database files with the `makedb` command. 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] makedb
```

You can see the state of the client distribution files by executing the `checkstatus` command (see [Section 8.5.4, "Check the Status of OLI Clients"](#)).

8.5.2 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.

Note: During execution of this command, some setup dialogs will display to show information on the installation or download activities. This is part of the packaging and therefore should not be disturbed in any way even if it takes a few minutes.

Each client package is written to a subdirectory of the client; the subdirectory is defined in the `OLI_CDS_ROOT` parameter in the `oli.ini` file.

```
oli.bat [-g] package
```

After a client package is successfully processed, its status is changed to `PACKAGE`. You can see the state of the client distribution files by executing the `checkstatus` command (see [Section 8.5.4, "Check the Status of OLI Clients"](#)).

8.5.3 Clean Up the OLI Tables Before Executing OLI for Another Distribution

The `cleanup` command cleans the OLI tables. The `cleanup` command purges the OLI tables. Execute this command before executing OLI for another distribution. There is no need to execute `cleanup` if you are not going to execute OLI for another distribution.

```
oli.bat [-g] cleanup
```

8.5.4 Check the Status of OLI Clients

Check the status of OLI clients with the `checkstatus` command. The initial status of a client is `RESET`. After the first client is processed successfully by `makedb`, 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`.

```
oli.bat checkstatus
```

8.6 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.

To deploy on native Win32 client devices, perform the following steps.

1. After a successful server side Offline 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`.

Configure Security in Oracle Database Mobile Server

The following sections detail how to manage security in Oracle Database Mobile Server:

- [Section 9.1, "Security Enhancements"](#)
- [Section 9.2, "Which Password is Which?"](#)
- [Section 9.3, "Providing Security for the Mobile Client"](#)
- [Section 9.4, "Configuring for Secure Socket Layer \(SSL\) Communication"](#)
- [Section 9.5, "Providing Your Own Authentication Mechanism for Oracle Database Mobile Server"](#)
- [Section 9.6, "Using a Load Balancer"](#)
- [Section 9.7, "Using Security Manager"](#)
- [Section 9.8, "Using a Firewall Proxy or Reverse Proxy"](#)
- [Section 9.9, "Providing SSL Client Authentication with a Common Access Card"](#)
- [Section 9.10, "Security Warning for Demo Applications"](#)

Note: There is additional information about developing for security in Chapter 7, "Customizing Oracle Database Mobile Server Security" in the *Oracle Database Mobile Server Developer's Guide*.

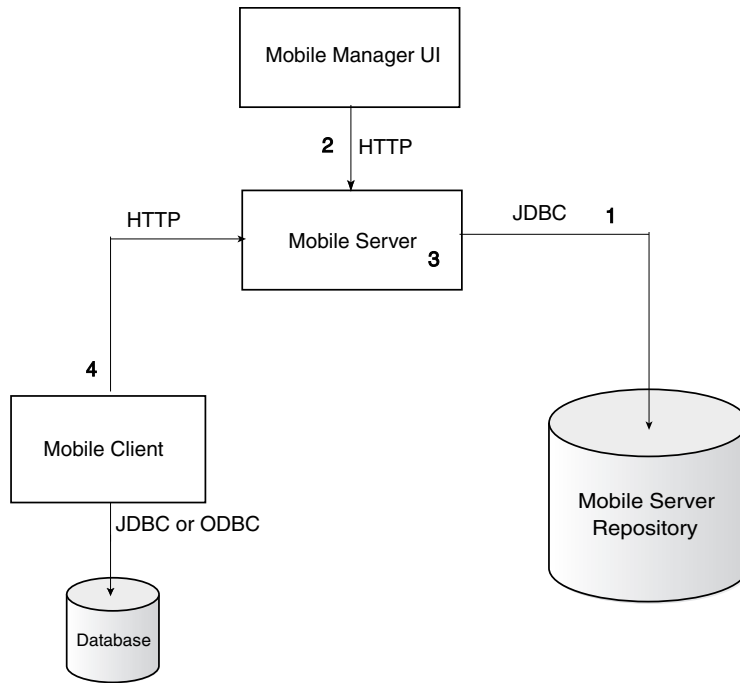
9.1 Security Enhancements

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.

9.2 Which Password is Which?

In the Oracle Database Mobile Server product, there are several user name/password combinations that are used for different security reasons and for separate types of users or administrators. This section describes each of the user name/password combinations in order to eliminate confusion.

Figure 9–1 Components That Use Passwords

As shown in [Figure 9–1](#), the passwords for the Database Mobile Server environment are as follows:

1. When the mobile server accesses the mobile server repository, it uses the repository user name/password. This defaults to the user `MOBILEADMIN` and the password is set during install.

Note: For details on how to modify the repository password, see [Section 9.2.1, "Modifying Repository Password"](#).

2. Within the mobile server, the following are two types of user name/password combinations, both of which are created using the Mobile Manager UI:
 - **Mobile Server Administrator:**
Only administrators can use the Mobile Manager, which is a Web administration UI that provides administrators the ability to modify how the mobile server behaves. The initial administration user name/password is created during installation. For adding or modifying, use the Mobile Manager to modify these on the mobile server.
 - **Mobile User:**
Normal mobile user provides its user name/password when synchronizing, which mobile server uses to verify that this user can access the application data. The mobile user name are stored in the mobile server repository. The password is stored in the repository only if external authentication is not used.

Note: See [Section 4.3.1.2.1, "Define User Name and Password"](#) for conventions for creating the user name or password.

- Modify the password on the client using the mSync UI. Only modify the password if you are connected to the mobile server to ensure that the user password change is propagated to the 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 user, then also send a Reset Password command from the Device Management section in the Mobile Manager to the mobile client. However, a Reset Password command can only modify the password if the client database is not encrypted. If the client database is encrypted, you can only modify the password with the mSync UI.

9.2.1 Modifying Repository Password

As described in number 1 in [Section 9.2, "Which Password is Which?"](#), the repository has its own user name/password combination. By default, the mobile server schema is MOBILEADMIN. However, if, during installation, you chose another name, replace MOBILEADMIN by your own schema name in the steps described in this section.

If you want to modify the password for your repository schema, perform the following:

1. Connect to your Oracle database where the mobile server repository was installed.
2. Change the password on the database with the ALTER command. The following command modifies the repository password to TEST:

```
alter user mobileadmin identified by TEST account unlock;
```

3. Backup the mobile.ora file on the mobile server:

```
cp $MOBILE_HOME/Mobile/Server/bin/mobile.ora $MOBILE_
HOME/Mobile/Server/bin/mobile.ora_backup
```

4. Comment out or remove ADMIN_USER and ADMIN_PASSWORD parameters in the mobile.ora file.
5. Restart the mobile server.
6. Go to the following URL: `http://mobileserver_hostname:Port/mobile/console/startup`
7. Log into the mobile server using the new repository password. For our example, this password is TEST.
8. Click "Save" to save the mobile server repository user name and password.

Note: This enables the auto start feature of the mobile server.

9.3 Providing Security for the Mobile Client

Security for the mobile client involves securing access to the device through authentication and securing the data stored within the mobile client database. For full details on both issues, see [Section 3.8, "Providing Security for the Mobile Client"](#) in the *Oracle Database Mobile Server Mobile Client Guide*.

9.4 Configuring for Secure Socket Layer (SSL) Communication

Oracle Database Mobile Server supports Secure Socket Layer (SSL) communication between the mobile server and mobile clients. 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 mobile server.

- [Section 9.4.1, "Creating an SSL Certificate"](#)
- [Section 9.4.2, "Configuring Mobile Server for SSL"](#)
- [Section 9.4.3, "Troubleshooting Error Messages for an SSL-Enabled Mobile Server"](#)
- [Section 9.4.4, "Support for Non-SSL Mobile Clients"](#)

Note: These are server-level steps which are typically executed prior to deployment of an application that requires SSL communication.

9.4.1 Creating an SSL Certificate

SSL communicates by validating an SSL certificate between the client and the server. This section describes how to create 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.

To create a keystore file, perform the following steps:

1. Use the Oracle 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 Oracle JDK. See <http://download.oracle.com/javase/6/docs/technotes/tools/index.html#security> 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, a message pops up asking if you want to continue. If you click Yes, a parameter is added to the `devmgr.ini` that tells Oracle Database Mobile Server to not validate the certificate. However, if you install the mobile client using any other method, you need to either set the SSL disable parameter yourself or add the certificate in the trusted certificate list. These options are listed below:

- To disable the SSL check, set the SSL disable parameter: Set the `DISABLE_SSL_CHECK` parameter to true in the `devmgr.ini` file to 1 after the `[NETWORK]` section, as follows:

```
[NETWORK]
DISABLE_SSL_CHECK=true
```

- To allow this certificate to be validated, add or install the new certificate in the trusted certificate list. After completion, the certificate will be trusted.

1. Open your browser and point it at the following site:

```
https://<server>/mobile/setup
```

Security alert displays.

2. Click "View Certificate".
3. Click "Install Certificate".
4. Click "Next-> Next-> Finish".

When you next perform the synchronization, the synchronization is successful. This adds the Oracle Database Mobile Server in the trusted root certification authority.

Note: If this is for testing only, then remove this certificate after testing is completed.

Each certificate authority has its own process for requesting and receiving signatures. Since this is outside the scope and control of Oracle Database Mobile Server, it is not covered in Oracle Database Mobile Server documentation. For more 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/
```

9.4.2 Configuring Mobile Server for SSL

Once you have a certificate, you must configure SSL on both the application server and the mobile server, as follows:

1. Configure SSL in the application server as documented in your application server documentation.
2. Configure SSL in the mobile server by adding `SSL=YES` and specifying `PORT=server_https_port` in the `[MOBILE]` section of the `mobile.ora` file. In the Mobile Manager, migrate to the "Administration" tab and select "Edit Config File". This is the `mobile.ora` file.
3. After all configuration is complete, restart the application server to initialize the changes.

Note: If you have both SSL and non-SSL clients wanting to access the mobile server, you must configure it to be both SSL and non-SSL enabled.

9.4.3 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.

9.4.4 Support for Non-SSL Mobile Clients

You may have a non-SSL mobile client that you want to interact with your SSL enabled mobile server. For the case when you have both SSL and non-SSL mobile clients interacting with a mobile server, the mobile server must be configured in both SSL and non-SSL mode. See [Section 9.4.2, "Configuring Mobile Server for SSL"](#) for details.

9.5 Providing Your Own Authentication Mechanism for Oracle Database Mobile Server

You can provide an external authenticator for the mobile server to authenticate users with passwords as well as their access privileges to applications. For example, in an enterprise environment, you may have your user data, such as employee information, stored in a LDAP-based directory service. The mobile server can retrieve the user information from the LDAP directory—or from any custom User Management System—if configured with your own implementation of an external authenticator. The mobile server links the external user information to the mobile server repository.

For full details, see Section 7.1, "Providing Your Own Authentication Mechanism for Authenticating Users for the Mobile Server" in the *Oracle Database Mobile Server Developer's Guide*.

9.6 Using a Load Balancer

Oracle Database Mobile Server supports handling client synchronization using mobile server cluster. To use this feature, multiple mobile servers could be set up behind a load balancer. The mobile client only needs connectivity to the load balancer machine, and when the load balancer receives an HTTP request, it will relay the client request to one of the clustered mobile server and send back any response data from the mobile server to the client. The load balancer will balance all the requests among the multiple mobile servers hence the overall throughput of the data synchronization will be increased.

- [Section 9.6.1, "Configure the Oracle Web Cache as a Load Balancer"](#)
- [Section 9.6.2, "Configure Mobile Server for Load Balancer"](#)
- [Section 9.6.3, "Communicate with Mobile Servers"](#)

9.6.1 Configure the Oracle Web Cache as a Load Balancer

Load Balancer is the host machine which has connectivity to the client machine. No mobile server should be installed on that machine. This release of Oracle Database Mobile Server supports load balancer using the Oracle Web Cache component, which is provided with *Oracle Fusion Middleware 11g Release 1*. For information on how to set up Oracle Web Cache as a load balancer, refer to the *Oracle Fusion Middleware Web Cache* documentation.

You must install mobile server cluster and Oracle Web Cache separately. Multiple mobile server instances in the cluster must share the same repository. After the installation of mobile server cluster and Oracle Web Cache, perform below configuration on the Oracle Web Cache to set up the load balancer. Refer to *Oracle® Fusion Middleware Administrator's Guide for Oracle Web Cache 11g Release 1 (11.1.1)* for more information.

1. Configuring Origin Servers with Oracle Web Cache

Go to "Oracle Web Cache Origin Servers" page. Add each mobile server in the cluster as an origin server. Specify the communication protocol as HTTP or HTTPS.

2. Configuring Site Definitions with Oracle Web Cache

Go to "Oracle Web Cache Site Definitions" page. Check if there is any web cache site. By default, the installation of Oracle Web Cache will create at least one site using HTTP protocol. If you want to configure the mobile client to communicate with the mobile servers using HTTPS protocol, add an Oracle Web Cache site using HTTPS.

Note: Refer to the Oracle Web Cache listening ports about which port should be used for HTTPS.

3. Configuring Site-to-Server Mapping with Oracle Web Cache

Go to "Oracle Web Cache Site-to-Server Mapping" page, pick up one Oracle Web Cache site and select all the mobile servers as origin servers for the site.

Note: If the Oracle Web Cache site is using HTTP protocol, the mobile server should also use HTTP protocol; likewise for HTTPS.

4. Configuring Session Binding on Oracle Web Cache

Go to "Oracle Web Cache Session Binding" page. For each of the Oracle Web Cache site that maps to the mobile servers, pick up option "Bind using a defined session", and specify "JSESSIONID" as Session Name, and "Cookie Based" as Session Binding mechanism.

9.6.2 Configure Mobile Server for Load Balancer

For each mobile server instance in the cluster, specify `DM_AUTO_SYNC_CACHE=YES` in `mobile.ora`.

9.6.3 Communicate with Mobile Servers

Use the IP address and port number of the Oracle Web Cache site as the URL to access mobile manager, download setup program for mobile client install, and synchronize client data. Users should never directly access the clustered mobile server URLs.

9.7 Using Security Manager

The Java Security Manager can be used with the application server to provide additional protection for resources running in the same Java Virtual Machine (JVM). Using a Java Security Manager is an optional security measure.

This section introduces how to use the Java Security Manager with the WebLogic Server to protect DMS resources by preventing untrusted code from performing actions against DMS that are restricted by the Java security policy file. You can also use the Java Security Manager with other supported application servers to protect DMS resources. Find more information about using the Java Security Manager with WebLogic here: http://docs.oracle.com/middleware/1213/wls/SCPRG/server_prot.htm#SCPRG392

- [Section 9.7.1, "Enabling Java Security Manager with WebLogic Server"](#)
- [Section 9.7.2, "Editing Java Security Policy File"](#)

9.7.1 Enabling Java Security Manager with WebLogic Server

To use the Java Security Manager with the WebLogic Server, specify the `-Djava.security.policy` and `-Djava.security.manager` arguments when starting the WebLogic Server. The `-Djava.security.policy` argument specifies a filename, using a relative or fully-qualified pathname, that contains Java security policies.

In a DMS environment, to enable the Java Security Manager with the WebLogic Server, add the `-Djava.security.manager` argument to the end of the `startWebLogic.sh` script file in `runmobileserver` on Linux, and `runmobileserver.bat` on Windows, and start DMS. For example:

```
#{WLS_DOMAIN_DIR}/bin/startWebLogic.sh -Djava.security.manager
```

where, `{WLS_DOMAIN_DIR}` is the domain directory of DMS in WebLogic, and the sample Java security policy file provided by WebLogic Server will be used.

9.7.2 Editing Java Security Policy File

WebLogic Server provides a sample Java security policy file, which you can edit and use. The file is located at `<WL_HOME>\server\lib\weblogic.policy`, where `<WL_HOME>` is the top-level directory of your WebLogic Server installation. Edit this policy file to add permissions for DMS resources.

- Add permissions for DMS to use JDBC Driver with WebLogic.


```
grant codeBase "file:<WL_HOME>/oracle_common/modules/oracle.jdbc_12.1.0/-" {
  permission javax.management.MBeanServerPermission "*";
  permission javax.management.MBeanTrustPermission "*";
};
```
- Add permissions for DMS to access system temporary files and DMS files to the section which is designated for all domains.


```
permission java.io.FilePermission "<<ALL FILES>>", "read";
permission java.io.FilePermission "/tmp", "write";
permission java.io.FilePermission "/tmp/-", "write";
```

```

permission java.io.FilePermission "<MOBILE_HOME>/mobile/server/ConsLog/*",
"write,delete";
permission java.io.FilePermission "<MOBILE_
HOME>/mobile/server/bin/mobile.ora1", "write";
permission java.io.FilePermission "<MOBILE_HOME>/mobile/server/bin/mobile.ora",
"write,delete";
permission java.io.FilePermission "<MOBILE_
HOME>/mobile/server/bin/resumefile.dat", "write,delete";
permission java.io.FilePermission "<MOBILE_
HOME>/mobile/server/admin/repository/mobile/console/cabo/images/-", "write";
permission java.lang.management.ManagementPermission "control";
permission java.net.SocketPermission "*", "accept,connect,resolve,listen";
permission java.lang.RuntimePermission "*";
permission java.sql.SQLPermission "*";

```

9.8 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.

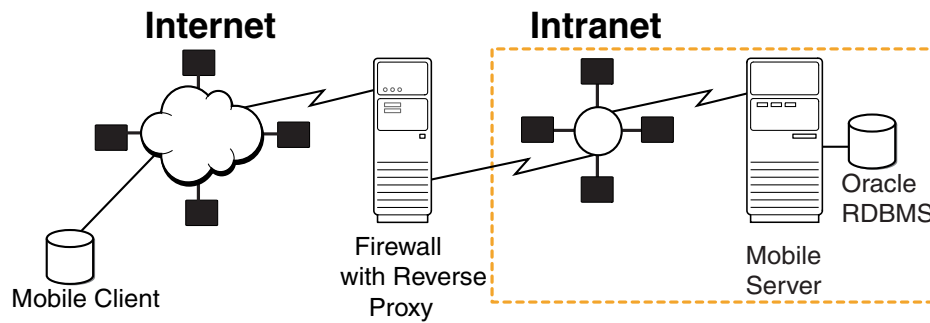
One can think of many scenarios in which case mobile users want to connect to the corporate server, without being directly connected to the corporate intranet. The corporate firewall uses proxy support to allow the mobile user to connect to the server. The following sections describe how to configure the mobile server and mobile client to communicate through a firewall:

- [Section 9.8.1, "Using a Reverse Proxy to Communicate from Internet to Intranet"](#)
- [Section 9.8.2, "Using HTTP Proxy to Communicate From Inside a Firewall"](#)

9.8.1 Using a 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 9–2](#). 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 9–2 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: Reverse proxy can be used with or without authentication. Using authentication is recommended for better security and access control to the inside of the firewall. The authentication is supported only if mobile client's user name and password are identical to those of the reverse proxy.

In order for this communication to occur seamlessly, do the following:

- [Section 9.8.1.1, "Configure the Apache Web Server as a Reverse Proxy"](#)
- [Section 9.8.1.2, "Set Up Mobile Server for Mobile Client Download"](#)
- [Section 9.8.1.3, "Download Reverse Proxy Mobile Client"](#)
- [Section 9.8.1.4, "Enable SSL When Using a Reverse Proxy"](#)
- [Section 9.8.1.5, "Configure Device Management to Work With a Firewall"](#)

9.8.1.1 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 user name of the proxy digest.
5. If you are using authentication, then configure the Reverse Proxy with the user name/passwords for all mobile clients that will access this reverse proxy for synchronization.

When the mSync is launched, the user name/password is sent automatically to the reverse proxy for authentication; thus, if the reverse proxy is not configured with the user name/password, then the connection is refused.

9.8.1.2 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, 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. Configure the mobile server to accept communication from the reverse proxy.

Configure the `reverse_proxy` parameter in the `mobile.ora` configuration file on the mobile server, as follows:

```
[MOBILE]
REVERSE_PROXY=http://<reverse_proxy_hostname>:<port_number>/mobile
```

2. If the reverse proxy is set up with authentication enabled, configure `HTTP_AUTH=YES` in `dmc.inf`

Add the following in INI element in

```
MOBILE_HOME\Mobile\Server\admin\repository\setup\dmc\dmc.inf,
where MOBILE_HOME is where you install the mobile server:
```

```
<item name="DEVGR" section="NETWORK">
<item name="HTTP_AUTH">YES</item>
</item>
```

For example, the updated INI element would be:

```
<ini>
<item name="DEVGR" 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>
<item name="DEVGR" section="NETWORK">
<item name="HTTP_AUTH">YES</item>
</item>
</ini>
```

9.8.1.3 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 9.8.1.2, "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, you must modify its configuration. Modify the `devmgr.ini` configuration file for your mobile client. Change or add the `SERVER_URL` parameter in the `NETWORK` section of the

devmgr.ini configuration file to point to the host/port of the reverse proxy server, as follows:

```
SERVER_URL=HTTP://<reverse_proxy_host>:<port>/mobile
```

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 7 and Windows Mobile devices—then Oracle Database Mobile Server 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 Mobile Server clients may stop working. This is critical if there are errors in certificate chaining. See [Section 9.8.1.4, "Enable SSL When Using a Reverse Proxy"](#) for details.

You may get "Unsuccessful http response: HTTP/1.1 408 Request Time-out" error in bglog log0.xml. This is caused by an open bug for Apache Http server.

9.8.1.4 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 user name/password for authentication. However, with msync, a browser is not displayed. Instead, msync sends on the user name/password for the user to the reverse proxy. Thus, you must have your environment configured correctly or the connection fails.

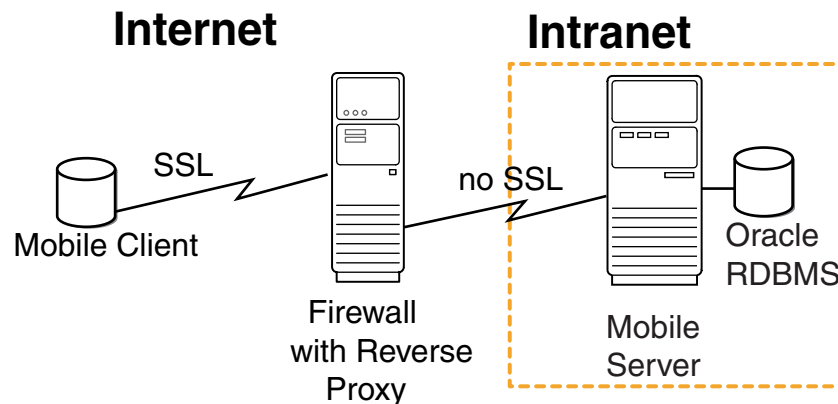
The following describes several scenarios that you may have between the mobile client and the reverse proxy:

- [Section 9.8.1.4.1, "Using SSL Authentication"](#)
- [Section 9.8.1.4.2, "Using SSL Between Mobile Client and Reverse Proxy"](#)
- [Section 9.8.1.4.3, "Using SSL Between Firewall and Mobile Server"](#)
- [Section 9.8.1.4.4, "Using Certificates That Are Not Signed By Trusted Authority"](#)

9.8.1.4.1 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.

9.8.1.4.2 Using SSL Between Mobile Client and Reverse Proxy As [Figure 9–3](#) demonstrates, you may want to encrypt your data and authenticate using SSL when using a reverse proxy.

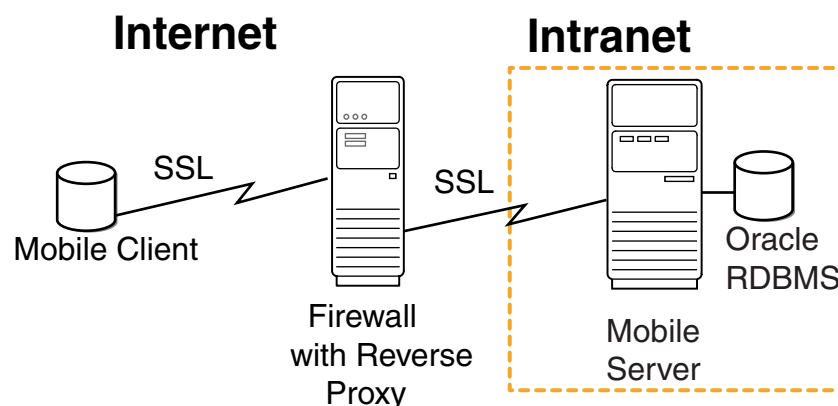
Figure 9–3 Mobile Client Communicating Over SSL Through Firewall Using Reverse Proxy



In this case, you must install the SSL certificate on the firewall for the SSL handshake between the mobile client and the firewall. If you are using a certificate that is not signed by a trusted authority or if you want to disable SSL authentication, see [Section 9.8.1.4.4, "Using Certificates That Are Not Signed By Trusted Authority"](#).

9.8.1.4.3 Using SSL Between Firewall and Mobile Server As [Figure 9–4](#) 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 9–4 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.

In general, install the chained certificate on the firewall for the SSL handshake between the mobile client and the firewall.

9.8.1.4.4 Using Certificates That Are Not Signed By Trusted Authority You can use certificates that are not signed by a trusted authority, such as a self-signed certificate, on the mobile server. In this case, set the following parameter in the `NETWORK` section in the `devmgr.ini` (`devmgr.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.

9.8.1.5 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 Manager 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.
- **Oracle Database 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 9.8.1.5.1, "Configure Mobile Device Listening Port"](#)
- [Section 9.8.1.5.2, "Firewall Configuration"](#)
- [Section 9.8.1.5.3, "Proxy Configuration for the Mobile Server"](#)

9.8.1.5.1 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.

9.8.1.5.2 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.

9.8.1.5.3 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 an 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:

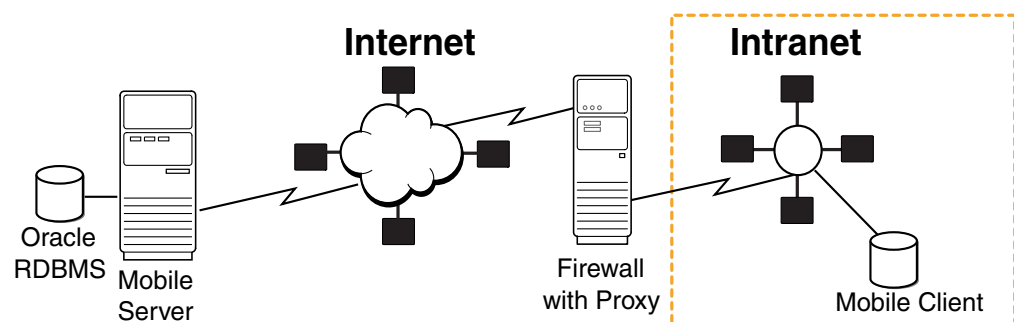
```
TIMEOUT=30&PROXY=http://proxy.foo.com:8080
```

9.8.2 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 9-5](#), 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 9-5](#), 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 9-5 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 9.8.2.1, "Proxy Configuration for Mobile Clients"](#)
- [Section 9.8.2.2, "Proxy Configuration for the Device Manager Agent"](#)
- [Section 9.8.2.3, "Reverse Proxy Configuration for HTTP PUSH from Mobile Server Not Supported"](#)

9.8.2.1 Proxy Configuration for Mobile Clients

For all 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.

9.8.2.2 Proxy Configuration for the Device Manager 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 `devmgr.ini` file, as follows:

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

9.8.2.3 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.

9.9 Providing SSL Client Authentication with a Common Access Card

Oracle Database Mobile Server supports client authentication for synchronization of data over SSL-based connections between a mobile client and the mobile server. The client authentication is based on X.509 certificates that can be stored in a Common Access Card (CAC).

Note: Encryption of client data with a CAC for databases on either mobile client is not supported.

The following sections describe how to enable client authentication in Oracle Database Mobile Server for Common Access Cards:

- [Section 9.9.1, "Introduction to SSL Client Authentication"](#)

- [Section 9.9.2, "Smartcard and Common Access Card Overview"](#)
- [Section 9.9.3, "Oracle Database Mobile Server Supports Common Access Cards"](#)
- [Section 9.9.4, "Supported Platforms for the Common Access Card"](#)
- [Section 9.9.5, "Prerequisites for Common Access Card"](#)
- [Section 9.9.6, "Configuration for Client Authentication Using the Common Access Card"](#)
- [Section 9.9.7, "Using the Common Access Card"](#)

9.9.1 Introduction to SSL Client Authentication

Secure Sockets Layer (SSL) is a security protocol that enables secure and authenticated data transfer over a network between a server and a client. During the connection setup, the server is always authenticated using a public key certificate. The client connects to the server only after establishing its identity based on the certificate.

The server may also choose to authenticate the client by asking for a client certificate. With client authentication enabled, the server requests a certificate from the client to verify that the client is who it claims to be. The certificate that the client sends must be an X.509 certificate and signed by a Certifying Authority (CA) that is trusted by the server. The server allows the connection if the client's certificate can be trusted.

9.9.2 Smartcard and Common Access Card Overview

SSL Client authentication requires that all users have their own X.509 certificate. There are several methods for users to store and manage certificates. One method is to store the certificates on a Common Access Card.

A Smartcard is a pocket-sized card with embedded integrated circuits that store and process data. A Common Access Card (CAC) is a Smartcard with additional software that enables you to securely store and manage security certificates and keys on the card, which are used for authentication. The data on the CAC is secured by a password or pin number. When the user connects to a server or system that requires authentication, the user is prompted for the password. The certificates and keys residing on the card are used to authenticate the user.

The CAC is used to authenticate users for access to computers and facilities. The CAC also enables encrypting and cryptographically signing email, uses PKI authentication tools, and establishes identity credentials. The certificates are stored in X.509 format on the CAC, and can be accessed by software running on a host machine through a card reader.

For example, the following describes what happens if a user browses a Web site that requires the client side certificate:

1. Open a browser and navigate to the Web site. The browser prompts for the pin number of the CAC.
2. After the user provides the correct pin, a CAC session is created. This enables all applications running on the machine to access the card and read the necessary certificates and keys. In case there is more than one certificate on the card that the server will accept, then the user is prompted to select which certificate to use.
3. By default, the CAC session expires after a configurable amount of time, after which the user must provide the pin number again.

For Oracle Database Mobile Server, the CAC is used to authenticate the client to the mobile server when requested before synchronization or any other type of communication between the client and the mobile server.

9.9.3 Oracle Database Mobile Server Supports Common Access Cards

Oracle Database Mobile Server supports client authentication over SSL connections using client side certificates and keys residing on the CAC. Oracle Database Mobile Server supports the ActivIdentity card. When the user supplies the CAC, the user also enters a pin number to retrieve the certificate and corresponding private key from the CAC. If the pin number is correct, data on the card can be accessed for a configurable period of time, known as the idle timeout that is set in the CAC software.

Note: To set the idle timeout for the authenticated sessions, use the User Console in the ActivIdentity software. The idle timeout defaults to 15 minutes.

When the pin number is verified, then access to the certificates and keys on the card is provided either to only the process that prompted the user to enter the pin number or to all of the user's processes running on the machine. For all components of the mobile client to work properly, the authentication must be shared among processes. See [Section 9.9.6.2.1, "Sharing Authentication Acceptance Across Processes"](#) for details.

9.9.4 Supported Platforms for the Common Access Card

The client authentication feature enabled for Oracle Database Mobile Server uses the CAC, which is supported only on the Win32 platform.

Note: At this point, client authentication is not supported on Windows Mobile or any UNIX platforms.

The components that must use the CAC when client authentication is requested are those that communicate with the mobile server. These are as follows:

- Mobile client
- Device Manager Agent
- Automatic Sync agent
- Packaging Wizard on Windows platform

MDW is not supported. Since MDW does not support SSL/HTTPS, it cannot provide client authentication over SSL.

9.9.4.1 Support for Mobile Clients That Are Not Enabled for Client Authentication

A mobile server that is configured for client authentication will only interact with mobile clients that are enabled for client authentication. Enabling client authentication on the mobile server is optional. Thus, if you do not turn it on, both clients with and without CAC can communicate with the mobile server without having to present a certificate.

9.9.5 Prerequisites for Common Access Card

You must install the software provided by the Common Access Card vendor to access the data on the card on each mobile client. This is required in order to access keys and certificates on the CAC. Refer to the vendor documentation on how to install this software.

9.9.6 Configuration for Client Authentication Using the Common Access Card

The following sections describe how to configure for client authentication:

- [Section 9.9.6.1, "Configuration of the Mobile Server to Request Client Authentication"](#)
- [Section 9.9.6.2, "Configuration of the Mobile Client to Use a CAC"](#)
- [Section 9.9.6.3, "Configuration for Reverse Proxy and Load Balancer"](#)

9.9.6.1 Configuration of the Mobile Server to Request Client Authentication

To configure the mobile server to request client authentication, perform the following:

1. Configure for SSL on the mobile server as described in [Section 9.4, "Configuring for Secure Socket Layer \(SSL\) Communication"](#).
2. Enable client authentication and install the certificate on the application server.
3. For the SSL connection to be established, the mobile server must have the certificate of the signing authority (CA), which signed the certificates present in the CACs. Import the CA certificate in the same keystore that you have used for storing the server certificate, as described in [Section 9.4, "Configuring for Secure Socket Layer \(SSL\) Communication"](#).

The following is an example of how to import the certificate using the `keytool` executable:

```
keytool.exe -keystore <samplekeystore> -storepass <oracle>
  -import -alias clientCACert -file <CACert.crt>
```

4. Modify the `pkcs11.cfg` file on the mobile server before you install the client with the correct path to the CAC library on the client. The PKCS11 configuration file defaults with the following entry:

```
library = C:\WINDOWS\system32\acpkcs11.dll
```

Note: Verify that this entry refers to the correct PKCS11 library provided by the CAC vendor on the client.

5. Restart the mobile server to reinitialize with the new modifications.

9.9.6.2 Configuration of the Mobile Client to Use a CAC

If you have configured mobile server to request client authentication, as described in [Section 9.9.6.1, "Configuration of the Mobile Server to Request Client Authentication"](#), then any `setup.exe` downloaded for a new mobile client automatically installs a mobile client enabled for handling certificates and keys from a CAC.

However, if the PKCS11 library location is different on the client than what is specified in the `pkcs11.cfg` file on the mobile server, as described in [Section 9.9.6.1, "Configuration of the Mobile Server to Request Client Authentication"](#), then modify the PKCS11 configuration file on the client to point to the correct directory.

The configuration file location is specified with the `PKCS11_CONFIG_FILE` parameter in the `%INSTALL_DIR%\bin\mobile.ora` file. The default configuration file is `%INSTALL_DIR%\bin\pkcs11.cfg`. Change the library property in the configuration file to refer to the correct library, as follows:

```
library = <Full path to PKCS11 library>
```

9.9.6.2.1 Sharing Authentication Acceptance Across Processes Client authentication sessions can be shared across processes belonging to the same user. Thus, if one process prompts the user for the pin number to read data from the CAC, a second process can also access the CAC without providing the pin number again--as long as the session is not expired.

You can control whether authentication is provided for all processes for the user or not through the Pin Caching Service.

On the ActivIdentity card, perform the following:

1. Select the Pin Caching Service through the following pull-down: Tools -> Advanced -> Configuration -> PIN Caching Service.
2. Select NO in the "Allow per process PIN caching" field.

9.9.6.3 Configuration for Reverse Proxy and Load Balancer

Configuration for the reverse proxy and load balancer are described in [Section 9.8, "Using a Firewall Proxy or Reverse Proxy"](#). However, you must configure the reverse proxy and load balancer to require and accept the client certificates and to be able to validate the client's certificates. In addition, the reverse proxy and load balancer must create an SSL connection to the mobile server using the client authentication.

The mobile server must be configured to accept the client authentication from the reverse proxy or load balancer.

9.9.7 Using the Common Access Card

If no CAC session is established yet, then--in most cases--the user is prompted for the CAC pin number to create a CAC session. The cases where the user may not be prompted are for the background processes: the Device Manager agent (`dmagent`) and the Sync Agent, which is the agent for automatic synchronization.

The Oracle Database Mobile Server `dmagent` acts both as an HTTP client when communicating with the mobile server and as an HTTP server when receiving DM commands over HTTP. The `dmagent` supports SSL over HTTP when acting as a client and supports client authenticated SSL connections.

If the background processes—the `dmagent` and the Sync Agent—start when the CAC session is already available, then the connection succeeds. However, once the CAC session expires, then the following may occur:

- If the CAC card is in the card reader, then the background processes prompt for the user pin.
- If the CAC card is not in the card reader, then the processes fail silently and a log error is written into `cac_log.txt` file.

Note: The SSL certificate or keys should not be stored in the database or any other persistence location on the mobile client.

9.10 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 Mobile Server 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 4, "Installation of Oracle Database Mobile Server" in the *Oracle Database Mobile Server Installation Guide*.




This chapter details the types of reports that you can view about the mobile server environment:

- [Section 10.1, "Viewing System Status Reports for the Server"](#)
- [Section 10.2, "Viewing Active User Sessions"](#)

10.1 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 10-1](#) displays, the Summary page lists Database, JDK, and Operating System details.

Figure 10–1 Summary Page

ORACLE Database Mobile Server 12c   
 Mobile Manager Logout Help

Mobile Servers | Mobile Devices
[Mobile Server: bdbcn6:9091](#) >

Summary Page Refreshed Sep 17, 2014 3:02:23 PM

Database

Name	Value
Database Version	12.1.0.1.0
Database Name	orcl12c
JDBC Driver	Oracle JDBC driver
JDBC Version	11.2.0.1.0
Schema Name	DMS121_GFS4_0904

Java

Name	Value
Java Runtime Environment version	1.8.0_11
Java Runtime Environment vendor	Oracle Corporation
Java installation directory	/home/dms/jdk1.8.0_11/jre

Operating System

Name	Value
Operating System Name	Linux
Operating System Architecture	amd64
Operating System Version	2.6.32-100.34.1.el6uek.x86_64
Path	/home/dms/DMS121_CertTest/glassfish4/glassfish/lib:/usr/java/packa ges/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib
Classpath	/home/dms/DMS121_CertTest/glassfish4/glassfish/modules/glassfish.jar:/home/dms/DMS121_CertTest/glassfish4/glassfish/lib/monitor/fla shlight-agent.jar

[Logout](#) | [Help](#)

Copyright © 1997, 2014, Oracle. All rights reserved.

10.2 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 10–2 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 10–2 Sessions Page

Sessions Page Refreshed Oct 14, 2014 9:43:2

User Name	Created On	Last Accessed On
ADMINISTRATOR	Oct 13, 2014 3:19:32 AM	Oct 13, 2014 3:30:40 AM

iOS Device Management

This chapter covers the following sections broadly:

[Section 11.1, "iOS Device Management with MDM"](#)

[Section 11.2, "Server and Security Configuration for iOS Mobile Device Management"](#)

11.1 iOS Device Management with MDM

Mobile Device Management (MDM) is a feature of modern mobile platforms such as iOS and Android that allows managing mobile devices remotely from the server. Unlike previous device management solutions, MDM is built-in to the mobile operating system (OS) and does not require special device management application (like `dmagent`) running on the mobile device.

In addition, it provides a much broader spectrum of features than can be achieved by client-side device management application.

The following sections describe basics and features of the MDM architecture:

- [Section 11.1.1, "Introduction to iOS MDM"](#)
- [Section 11.1.2, "Profiles and Device Enrollment"](#)
- [Section 11.1.3, "iOS MDM Commands"](#)

11.1.1 Introduction to iOS MDM

See the following sections:

- [Section 11.1.1.1, "Mobile Server and MDM"](#)
- [Section 11.1.1.2, "MDM and Push Notifications"](#)

11.1.1.1 Mobile Server and MDM

As mentioned above, support for MDM is built-in to iOS operating system. iOS devices exchange messages with Mobile Server using special MDM protocol, which enables to retrieve device information, send commands to devices, configure devices with various rights and restrictions and manage mobile applications. Apple has supported MDM for a while, however many new features and changes to the protocol have been added. Currently, Mobile Server supports management of devices running iOS 7.

Mobile Server seamlessly integrates iOS devices into the existing mobile architecture. iOS devices are assigned to users and managed using Mobile Manager. iOS applications can be published using Packaging Wizard, and iOS applications running

on iOS devices can synchronize data stored in local sqlite or Berkeley DB database with Oracle back-end Database, using existing synchronization architecture.

However, some device management commands supported for other platforms are not yet supported for iOS. This includes synchronization-related commands.

For a high-level overview of iOS MDM, see the following:

<https://www.apple.com/iphone/business/it/management.html>

Note: Not all features described in the document (<https://www.apple.com/iphone/business/it/management.html>) are currently supported by Mobile Server.

In particular, we do not yet support Device Enrollment Program (where devices can be enrolled in MDM at device activation time) or Volume Purchasing Program (allowing purchasing of applications from iTunes Application Store).

For more information on MDM protocol and additional resources see [Appendix D.1, "Description - MDM Architecture"](#).

11.1.1.2 MDM and Push Notifications

Apple Push Notification Service is a cloud-based service run by Apple which is used to deliver short messages to iOS devices. The main purpose is to deliver notifications to iOS applications, but it is also a crucial part of MDM protocol to notify iOS devices that new commands are available (see [Appendix D.1, "Description - MDM Architecture"](#)). Devices listen to push notifications by having SSL connections to APNS servers. The details of these connections, as well as APNS servers themselves are managed by Apple, providing a seamless interface to the user. For more information about APNS, see:

<https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html>

Mobile Server sends push notifications to iOS devices by connecting to APNS Gateway and delivering appropriate message which will be used by APNS servers to deliver notification to the device. Usually push notifications are targeted for and processed by a particular application on the device. The target application is identified by a special string called push notification topic.

However, for MDM, push notifications are processed by iOS operating system MDM component. The topic string for MDM must start with a special prefix. For more information, see, [Section 11.2.3.7.1, "Push Notification Topic"](#).

Current address of production APNS gateway is **gateway.push.apple.com, port 2195**.

Note: Both Mobile Server and your managed iOS devices must make SSL connections to APNS gateway for MDM to function. If either your Mobile Server or your iOS devices are behind firewall, you may have problems with managing your iOS devices. To fix those, your network administrator must open up some ports in the firewall.

Refer to the Apple support note:

<http://support.apple.com/en-us/HT203609>

To cite this note, the following ports must be open:

- TCP port 5223 (used by devices to communicate to the APNs servers)
 - TCP port 2195 (used to send notifications to the APNs)
 - TCP port 2196 (used by the APNs feedback service)
 - TCP Port 443 (used as a fallback on Wi-fi only, when devices are unable to communicate to APNs on port 5223)
-
-

11.1.2 Profiles and Device Enrollment

To manage iOS device, you must enroll it in MDM. See [Section 7.11.1, "iOS Device Enrollment"](#) to see the enrollment steps from the user's perspective. During and after enrollment, you can customize settings on your devices by installing configuration profiles. In fact, the enrollment itself is done via installing a special configuration profile on the device. For more information on configuration profiles and what happens during enrollment process, see the sections below:

- [Section 11.1.2.1, "Configuration Profiles"](#)
- [Section 11.1.2.2, "Device Enrollment Process"](#)

11.1.2.1 Configuration Profiles

Configuration Profile is an XML file used to distribute iOS configuration information. It uses property list format (for information on property list format, see <https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/PropertyLists/Introduction/Introduction.html>). It allows you to specify restrictions on device features, WiFi settings, VPN settings, LDAP directory server settings and many others. In addition, certificates and keys can also be transmitted via configuration profiles. For full description of configuration profile capabilities and format, see *Configuration Profile Key Reference* here: <https://developer.apple.com/library/ios/featuredarticles/iphoneconfigurationprofile/introduction/introduction.html>

Note: Configuration profile files have extension **.mobileconfig**.

There are several ways to deploy configuration profiles to the devices. For example, a profile can be emailed to the device as an attachment. Once you open the attachment on iOS device, iOS will know that this is a configuration profile and will install it. However, the easiest way to deploy configuration profile is via MDM. Mobile Server, acting as MDM server, can push configuration profiles automatically to managed iOS devices.

Configuration profiles consist of one or several payloads. Usually there is a one main profile payload that contains other setting-specific payloads - other payloads being

embedded as an array inside **PayloadContent** element. Here is an example of the profile used to configure device passcode settings:

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>PayloadContent</key>
  <array>
    <dict>
      <key>PayloadDescription</key>
      <string>Configures Configuration Profile security</string>
      <key>PayloadDisplayName</key>
      <string>Profile Security</string>
      <key>PayloadIdentifier</key>
      <string>com.oracle.dms.devpass.ProfileSecurity</string>
      <key>PayloadIdentifier</key>
      <string>com.oracle.dms.devpass.ProfileSecurity</string>
      <key>PayloadOrganization</key>
      <string>Oracle</string>
      <key>PayloadType</key>
      <string>com.apple.profileRemovalPassword</string>
      <key>PayloadUUID</key>
      <string>D3150EF6-299B-4D63-B803-341B159348B2</string>
      <key>PayloadVersion</key>
      <integer>1</integer>
      <key>RemovalPassword</key>
      <string>oracle</string>
    </dict>
    <dict>
      <key>PayloadDescription</key>
      <string>Configures security-related items.</string>
      <key>PayloadDisplayName</key>
      <string>Passcode</string>
      <key>PayloadIdentifier</key>
      <string>com.oracle.dms.devpass.passcodepolicy</string>
      <key>PayloadOrganization</key>
      <string>Oracle</string>
      <key>PayloadType</key>
      <string>com.apple.mobiledevice.passwordpolicy</string>
      <key>PayloadUUID</key>
      <string>6B7812F6-C86C-4AAF-BE5E-CB7A587D38D7</string>
      <key>PayloadVersion</key>
      <integer>1</integer>
      <key>allowSimple</key>
      <true/>
      <key>forcePIN</key>
      <true/>
      <key>maxFailedAttempts</key>
      <integer>10</integer>
      <key>maxInactivity</key>
      <integer>5</integer>
      <key>maxPINAgeInDays</key>
      <integer>730</integer>
      <key>minLength</key>
      <integer>5</integer>
      <key>pinHistory</key>
      <integer>10</integer>
    </dict>
  </array>
  <key>PayloadDescription</key>
```



```

<string>Device passcode policy.
</string>
<key>PayloadDisplayName</key>
  <string>Device passcode profile</string>
  <key>PayloadIdentifier</key>
  <string>com.oracle.dms.devpass</string>
  <key>PayloadOrganization</key>
  <string>Oracle</string>
  <key>PayloadRemovalDisallowed</key>
  <true/>
  <key>PayloadType</key>
  <string>Configuration</string>
  <key>PayloadUUID</key>
  <string>D34F4145-4FE8-4A4B-A45C-5857D54EAB2D</string>
  <key>PayloadVersion</key>
  <integer>1</integer>
</dict>
</plist>

```

Note: The same profile can configure several (and could be un-related) settings via embedded payloads. The following tools automate creation of configuration profiles so that you do not have to write XML manually:

- Apple Configurator - For Mac OS 10.8 and later only.
You can download it from the AppStore, see <https://itunes.apple.com/us/app/apple-configurator/id434433123?mt=12>
 - iPhone Configuration Utility - Available for both Mac OS (10.8 and earlier), see <http://support.apple.com/kb/DL1465> and Windows, see <http://support.apple.com/kb/DL1466>.

iPhone Configuration Utility may not support the latest features of iOS 7. If you need full iOS 7 support, use Apple Configurator.
-

You can create your own configuration profiles to customize the settings on your managed iOS devices. You can then upload these profiles to the Mobile Server via Mobile Manager interface. For more information, see [Section 7.11.4, "iOS Profile Management on Mobile Manager"](#).

Note: All uploaded profiles will be deployed to all iOS devices when they enroll in MDM.

Configuration profiles can be either signed or unsigned. Signed profiles avoid having warning displayed on iOS device when the profile is installed. You do not need to sign configuration profiles that you upload to the Mobile Server (in any case iPhone Configuration Utility by default uses self-signed certificate which will not be trusted by the device). Mobile Server will sign the profiles automatically when they are uploaded.

MDM Certificate Authority (CA) is used for signing of the profiles. For more information on MDM CA, see [Section 11.2.1, "Creating and Configuring Certificate Authority"](#).

Another type of profile is provisioning profile. Provisioning profiles tell iOS device whether a given application can run and also what privileges that application may have. Provisioning profile files usually have extension **.mobileprovision**. Provisioning profiles can also be uploaded to the Mobile Server (see [Section 7.11.4, "iOS Profile Management on Mobile Manager"](#)) and will be deployed to all iOS devices that enroll in MDM. See [Section 11.1.4, "Managing iOS Applications"](#) for more details.

11.1.2.2 Device Enrollment Process

During enrollment process:

- Device receives Certificate Authority certificate that allows it to trust Mobile Server.
- Mobile Server uniquely identifies the device and stores this identification in repository.
- Device receives special configuration profile allowing it to receive and execute commands from Mobile Server.
- Mobile Server receives initial detailed information about the device that would be displayed in Mobile Manager (see [Section 7.11.2, "View iOS Device Information"](#)). You can refresh this information by using corresponding MDM commands (see [Section 11.1.3.2, "Commands"](#)).
- All uploaded configuration and provisioning profiles will be installed on the device.
- All published applications for chosen platform (iOS Berkeley DB or iOS SQLite) to which the user has access will be installed on the device.

See [Appendix D.2, "Device Enrollment Process,"](#) for more information on device enrollment process used by Mobile Server.

11.1.3 iOS MDM Commands

MDM protocol contains a rich set of commands (described in MDM Protocol Reference, see [Chapter D.1.2, "Additional Resources"](#)), not all of which are currently supported by Mobile Server. Here we will mention the commands that are currently supported. [Section 7.11.3.3, "View Command History,"](#) describes Mobile Manager interface for sending the commands to iOS devices.

The commands, by nature, are asynchronous. It may take some time between when the command is sent to the device and the status and the results are received back (though usually it happens very fast). When the status and results are received by Mobile Server, they will be stored in the repository and can be retrieved for display in Mobile Manager. You may have to wait and refresh "Device Info", "Software Info" and "Command History" tabs if you do not see the results right away.

You can find the current state of the command by looking at the command status. The following sections discuss commands and command status information:

- [Section 11.1.3.1, "Command Status"](#)
- [Section 11.1.3.2, "Commands"](#)

11.1.3.1 Command Status

You can find the status of each command on the Command History page in Mobile Manager. See [Section 7.11.3.3, "View Command History"](#) for more information on Command History page. Here we list command status values:

- **Command Queued** - Command has been queued in the Mobile Server repository command queue but has not been sent to the device yet.
- **Push Notification Sent** - Mobile Server has sent push notification to wake up the device and prompt it to poll the server for commands.
- **Command Sent** - Command has been sent to the device.
- **Command Succeeded** - Device successfully executed the command. Mobile Server has received the command status and results, if any, and stored them in the repository.
- **Command Failed** - Device has failed to execute the command and sent the error information back to the Mobile Server. Additional error information may be displayed.
- **Device Not Ready** - Sometimes device is unable to execute a command at a certain time, but the command can be executed at a future time. In this case, device sends a special **NotNow** status to the Mobile Server. Once device is ready to execute the command, it will poll the server again. This only applies to certain commands. Some commands can always be executed and are immune to the **NotNow** status. You can find more information on **NotNow** status in MDM Protocol Reference, see [Appendix D.1.2, "Additional Resources"](#). When describing the commands we will indicate which commands can always be executed.
- **Command Execution Started** - Not all commands are pure MDM commands (that is described by command cycle in [Appendix D.1.1, "MDM Command Cycle"](#)). There are commands that may send more than one MDM command to the device and also perform server-side logic in between. For example, "Update Application" command will first send "Retrieve Software Information" to get the current application version installed on the device. It will then compare retrieved version with current published version of the application on the server, and only if a later version is published, will send "Install Application" command to the device to install the new application version.

We call these commands "Composite Commands". Composite commands start with "Command Execution Started" status. The only other valid status values for composite commands are "Command Succeeded", "Up To Date", and "Command Failed".

Note: The Command History page in Mobile Manager will display both composite commands and constituent MDM commands that they invoked.

- **Up To Date** - This status is only valid for "Update Application" command and tells that application installed on the device is of latest version published, so no installation action was taken.
- **MDM Protocol Error** - Device has received malformed command. Contact Oracle Support for assistance. Additional error information may be displayed.
- **Failed to Prepare Command** - Mobile Server failed to prepare command payload. This usually indicates problem with Mobile Server repository. Contact Oracle Support for assistance. Additional error information may be displayed.
- **Failed to Process Result** - Mobile Server failed to process result payload from the device. This usually indicates problem with Mobile Server repository. Contact Oracle Support for assistance. Additional error information may be displayed.

11.1.3.2 Commands

The following commands are supported for iOS devices (see [Section 7.11.3, "iOS MDM Command Management"](#) on how to send commands using Mobile Manager interface):

- **Install Configuration Profile for iOS Device** - You can choose one of the profiles you uploaded (see [Section 7.11.4, "iOS Profile Management on Mobile Manager"](#)).

Note: All uploaded profiles will be installed on the device at enrollment time. You only need to use this command to install additional profiles uploaded after device has enrolled in MDM.

- **Remove Configuration Profile for iOS Device** - Removes configuration profile from the device (you can choose from the list of the profiles you uploaded).
- **Install Provisioning Profile for iOS Device** - You can choose one of the provisioning profiles you uploaded (see [Section 7.11.4, "iOS Profile Management on Mobile Manager"](#)).

Note: All uploaded provisioning profiles will be installed on the device at enrollment time. You only need to use this command to install additional provisioning profiles uploaded after device has enrolled in MDM.

- **Remove Provisioning Profile for iOS Device** - Removes provisioning profile from the device (you can choose from the list of the provisioning profiles you uploaded).
- **Device Information** - Once Mobile Server receives device information, it will be displayed on the "Device Info" page (see [Section 7.11.2, "View iOS Device Information"](#)). As mentioned above, you may have to refresh "Device Info" page if the updated information does not appear immediately.

Note: The device information will be collected from the device when it enrolls. You only need to use this command to get up-to-date information that may have changed (for example, battery level or free disk space).

- **Retrieve iOS Security Information** - This information is complimentary to the device information and will be displayed on the "Security" panel of "Device Info" page (see [Section 7.11.2, "View iOS Device Information"](#)). MDM protocol uses different command to retrieve this information (as opposed to "Retrieve Device Information"). The same considerations above apply to this command as to "Retrieve Device Information".
- **Lock iOS Device** - Locks device. Optionally, a message can be provided to be displayed on device screen when it is locked (it will only be displayed if device is protected by passcode). Otherwise default message will be displayed.

This command is helpful when the device gets lost. The message may indicate a contact information where to return the device.
- **Clear Passcode for iOS Device** - Clears passcode from the device.

Note: If a configuration profile with passcode requirement is installed on the device, the device will require a new passcode to be entered within 60 minutes.

- **Erase iOS Device** - Reset device to factory settings. All information on the device will be lost. This command is used when the device is lost or stolen. The device will need to re-enroll into MDM.
- **Retrieve Software Information** - Retrieve installed application information from the device, such as version, size, etc. You can either choose which application to retrieve for, or choose to retrieve for all applications.

Note: Only information for managed applications will be retrieved, that is only applications that were installed from Mobile Server via MDM (as opposed to applications installed by device's user from elsewhere). The information retrieved will be displayed on "Software Info" page in Mobile Manager (see [Section 7.11.2, "View iOS Device Information"](#)). As for "Device Info" page, you may have to refresh this page if the updated information does not appear immediately.

- **Retrieve iOS Application Status** - Retrieve application status for one or all applications. This information will also be displayed on "Software Info" page in Mobile Manager, in "Status" column. This command is used to get up-to-date installation status of the application (since some applications may take a while to install) and find if any problems have occurred during installation. Here is the excerpt from MDM Protocol Reference (see, [Chapter D.1.2, "Additional Resources"](#)) describing possible status values (we omitted values that are not relevant to enterprise-distributed applications):
 - **Installing** - The application is being installed.
 - **Managed** - The application is installed and managed.
 - **ManagedButUninstalled** - The application is managed, but has been removed by the user. When the application is installed again (even by the user), it will be managed once again.
 - **Updating** - The application is being updated.
 - **Unknown** - The application state is unknown.

The following statuses are transient, and are reported only once:

- **UserInstalledApp** - The user has installed the application before managed application installation could take place.
- **UserRejected** - The user rejected the offer to install the application.
- **UpdateRejected** - The user rejected the offer to update the application.
- **Failed** - The application installation has failed.
- **Install Application** - Installs one of the published applications on the device.

Note: If this command is issued for an application already installed on the device, it will be treated as an update. Application data should not be lost in this case.

- **Remove iOS Application** - Uninstalls application from the device. All application data on the device will be removed.
- **Update iOS Application** - This is a composite command. It will check what version of application is installed on the device and will compare it with the version published on Mobile Server.

If higher version is published (or if application is not installed on the device), "Install Application" command will be issued to update the application. Otherwise, the status of this command will be set to "Up To Date".
- **Update All iOS Applications** - Same as "Update Application" command, but does for all managed applications assigned to the device's user.
- **Un-enroll iOS Device from MDM** - Terminates device's enrollment so that the device becomes unmanaged. This is done via removing configuration profile with MDM payload that was installed on the device during enrollment (see [Section 11.1.2.2, "Device Enrollment Process"](#)).

Removing this profile will automatically remove all the configuration and provisioning profiles and all the applications installed by Mobile Server. All the information in the Mobile Server repository pertaining to this device will be removed.

11.1.4 Managing iOS Applications

This section describes concepts and issues related to managing iOS applications with Mobile Server.

- [Section 11.1.4.1, "Provisioning Profiles"](#)
- [Section 11.1.4.2, "Preparing and Publishing iOS Application"](#)
- [Section 11.1.4.3, "Application Security Considerations"](#)

11.1.4.1 Provisioning Profiles

To run on iOS devices, every application must be digitally signed. There are two kinds of signing certificates - for development and for production. Both can be obtained from Apple Developer's portal. Xcode creates certificate signing request and submits it to the portal. Once approved, the developer will have a signing identity to sign the application. The signing certificate itself is signed by Apple Certificate Authority, so it will be trusted by the device.

However, even when signed, Apple wants to restrict which devices can run the application. Otherwise, an application can avoid going through Apple AppStore approval process that verifies that the application is safe for anybody to use. That is where provisioning profiles come in.

Provisioning profile is an XML file in plist format (see [Section 11.1.2.1, "Configuration Profiles"](#)). These files have extension **.mobileprovision**. There are different provisioning profiles for development and production.

Development provisioning profiles are used for application development and testing. They ensure that the application can run only if it is built and signed by certain developers or team registered on the portal and only on certain devices which also have to be registered on the portal. Development provisioning profile contains:

- Application Identifier

This can contain wildcards so that it can match multiple application bundle ids. For example, if the application identifier is com.oracle.*, it will cover all applications for which the bundle id starts with com.oracle.

- List of developer certificates who are allowed to build and sign the application. Usually it includes all developers in the team.
- List of UDIDs of devices on which application is allowed to run.
- Other information like creation and expiration dates, and others.

An application will only run on the device if the provisioning profile is installed on the device and if all of the following is true:

- Application bundle id matches the profile's application identifier.
- Application is signed by one of the certificates listed in the profile.
- The device's UDID is listed in the profile.

You do not create provisioning profiles manually, instead they are created on the developer's portal (see [Section 11.1.4.2, "Preparing and Publishing iOS Application"](#)) from where you can download them. Each profile is signed by Apple. You can open **.mobileprovision** file in text editor (the binary data you would see around the text is the signature container in CMS format, see <http://tools.ietf.org/html/rfc5652>) to find more information.

Distribution provisioning profiles are used either for submission of the application to the AppStore or for the enterprise app distribution. We will not cover the former here since the later is what you need to publish and manage your application from the Mobile Server (see [Section 11.1.4.2, "Preparing and Publishing iOS Application"](#)).

Apple allows you to distribute your application to a limited audience in your Enterprise. Because of this, distribution provisioning profiles do not specify list of devices (you will see ProvisionsAllDevices key in the profile if you open it in text editor), which means that the application can run on all devices on which the provisioning profile is installed. In addition, the list of certificates in the profile usually contain only one certificate, the one with which the application was signed.

Note: You must be a team administrator in Enterprise Developer's Program to create distribution provisioning profiles (see [Section 11.1.4.2, "Preparing and Publishing iOS Application"](#)).

11.1.4.2 Preparing and Publishing iOS Application

Applications managed by MDM use Apple's Enterprise Distribution model. In this model, applications do not need to be submitted to Apple AppStore, but instead are distributed in-house to your employees and customers. This approach does not require Apple AppStore approval process and the applications are distributed to limited audience only instead of general public.

You must have Apple Enterprise Developer Account to create and distribute Enterprise Applications. You can apply for Enterprise Program here: <https://developer.apple.com/programs/ios/enterprise/>

For information to get started with Enterprise Program, see <https://developer.apple.com/programs/ios/enterprise/gettingstarted/>

To distribute iOS Application for enterprise you must obtain distribution certificate (signing identity) as well as distribution provisioning profile, as opposed to developer

certificate and developers provisioning profile. To get more information on this, you can use the following resources:

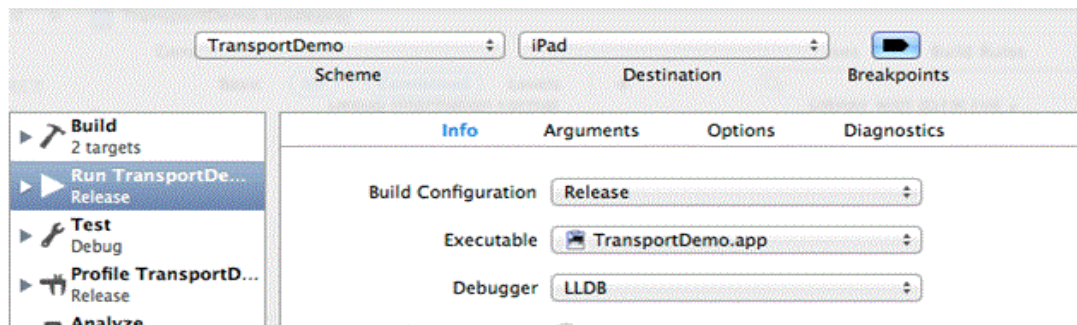
- Apple Application Distribution Guide
Contains a lot of information on managing and distributing applications, but targeted for AppStore distribution. See here:
<https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/Introduction/Introduction.html>
- Tutorial explaining enterprise in-house distribution. See here:
<http://johannesluderschmidt.de/provision-ios-ipa-app-for-in-house-enterprise-distribution/2993/>

Note: You must be a (or contact) team administrator to create distribution certificates and provisioning profiles.

The following steps help to build and distribute your enterprise application and publish it to the Mobile Server (based on sample "TransportDemo" application):

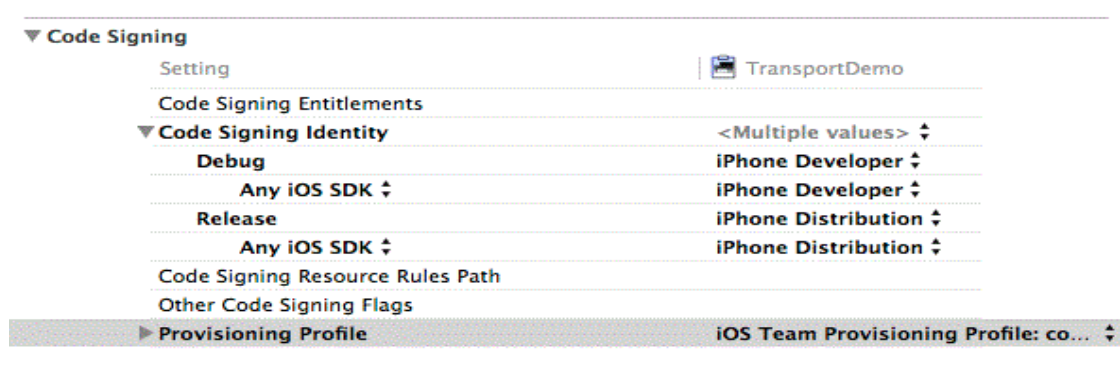
- Open your application project in Xcode (on your Mac machine). Ensure that you set your build scheme to build for iOS device and choose Release Configuration (you can do so from the menu *Product -> Scheme -> Edit Scheme*) (although you can use Debug Configuration as well if need arises). See [Figure 11-1](#).

Figure 11-1 Setting Release Build Configuration in Xcode



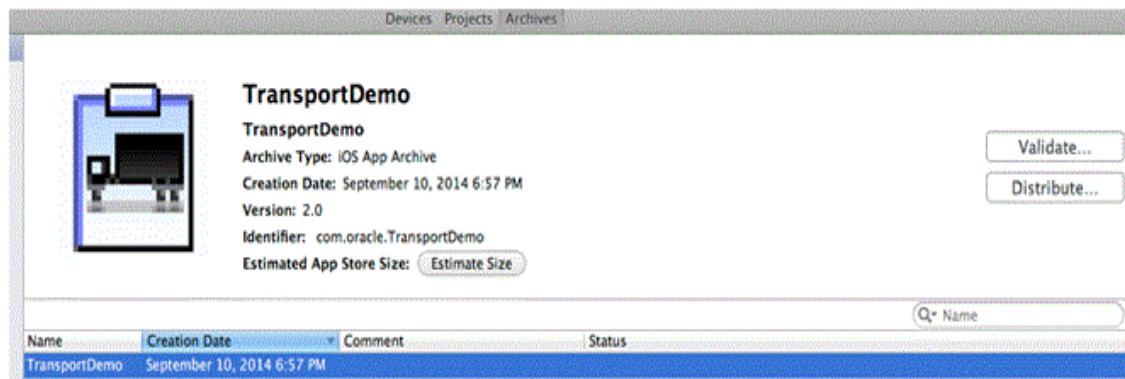
- Select your project in project navigator. Select your application target on the left and select "Build Settings" tab (in your application target build settings). Ensure to choose distribution code signing identity for Release build. See [Figure 11-2](#).

Figure 11–2 Setting Code Signing Identity in Xcode



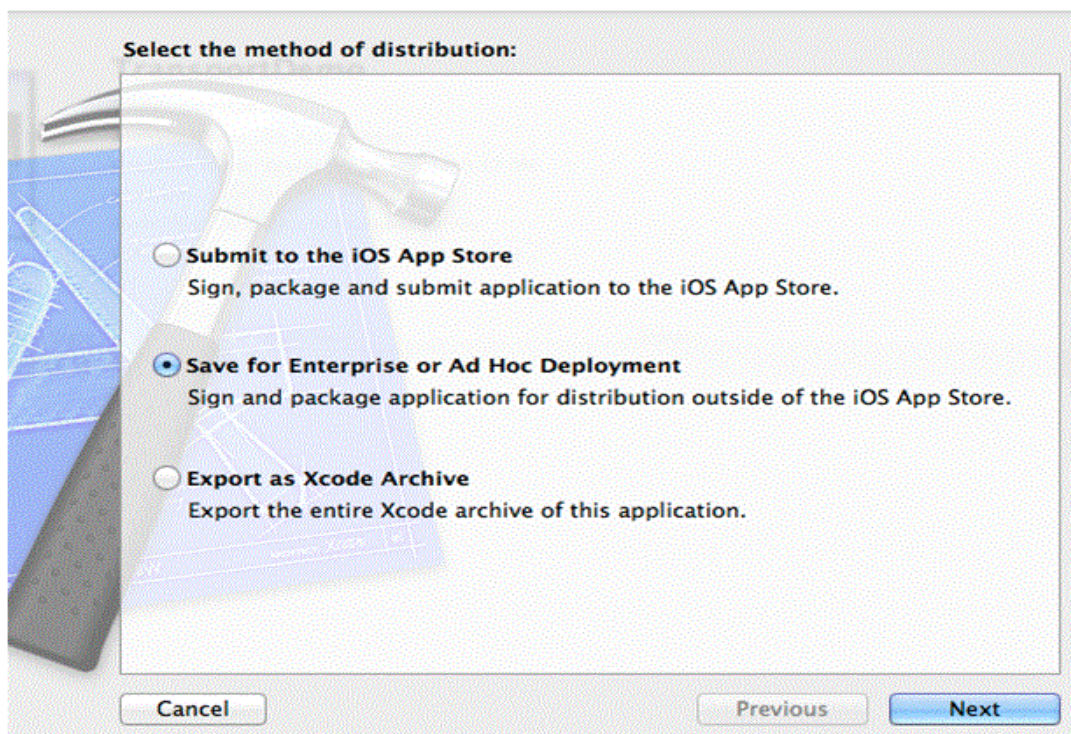
- Build your application (Cmd-B or from the Product menu).
- Go to menu *Product* -> *Archive*, after successful build. Click on "Distribute" See Figure 11–3.

Figure 11–3 Archiving your Application



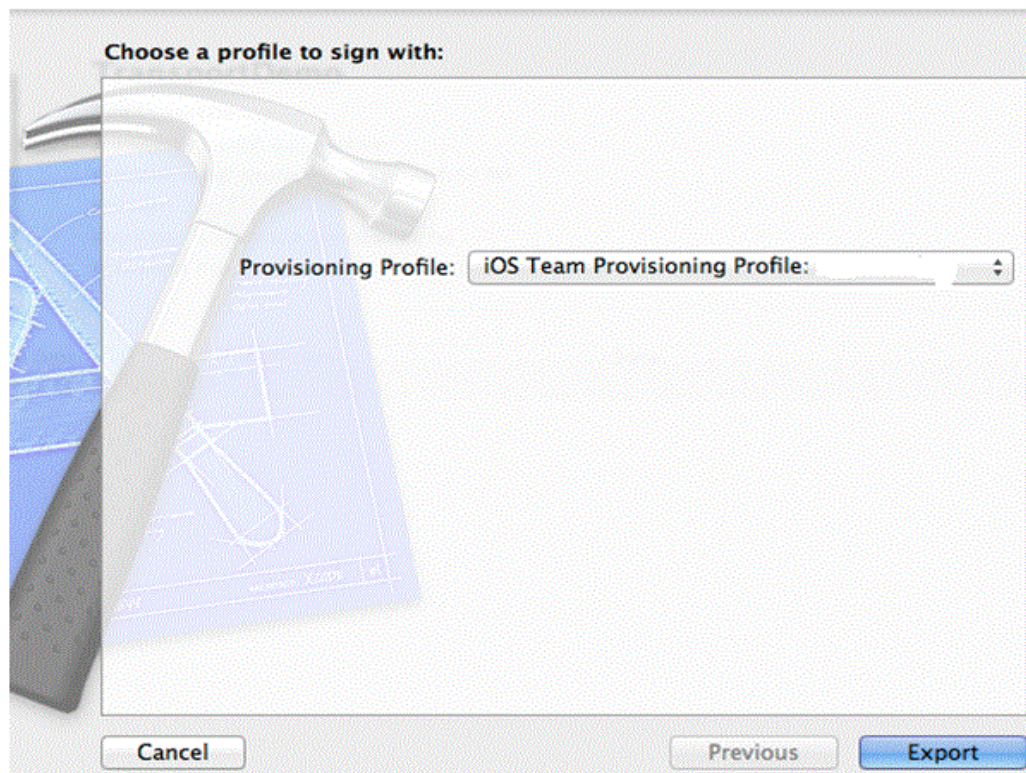
- Choose "Save for Enterprise or Ad Hoc Deployment" and click "Next". See Figure 11–4.

Figure 11-4 *Distributing your Application*



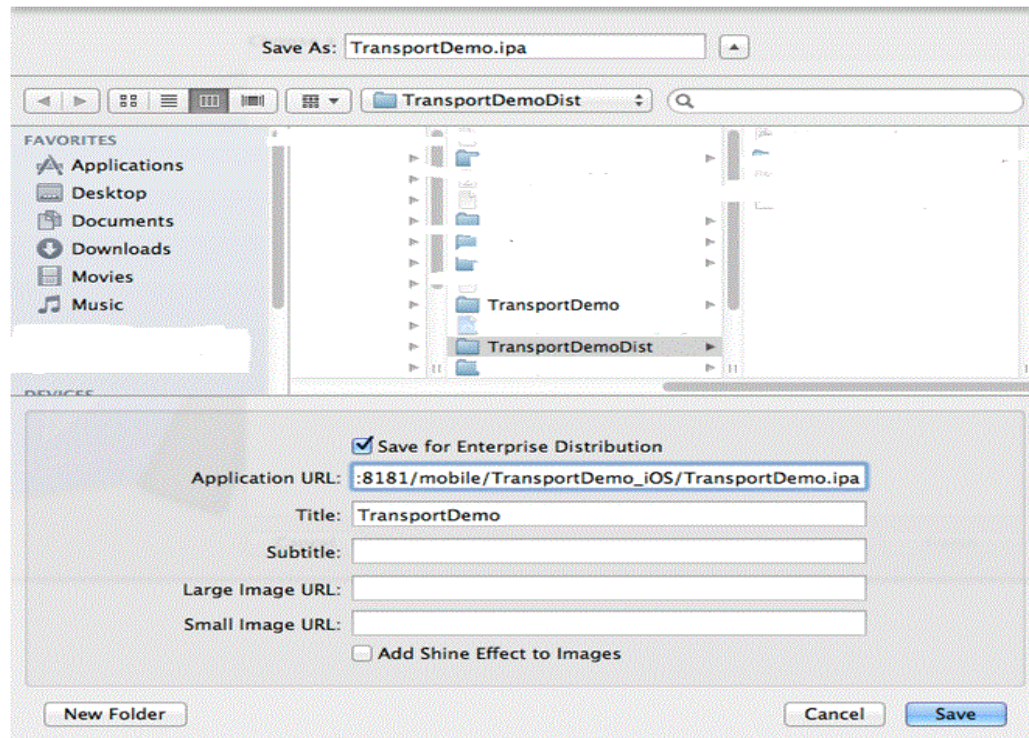
- Ensure your distribution provisioning profile is selected and click "Export". See [Figure 11-5](#). You will see a dialog box as in [Figure 11-6](#).

Figure 11-5 *Choosing Provisioning Profile for Distribution*



- Export your application as .ipa archive file. Choose the destination directory. Check the check-box "Save for Enterprise Distribution". You will be prompted to enter some information. See [Figure 11-6](#).

Figure 11-6 Exporting your Application



Application URL is the URL where your application will be hosted on Mobile Server. The URL should be in the following format:

https://mobile_server_host:mobile_server_port/mobile/app_virtual_path/app_name.ipa

where:

- mobile_server_host** and **mobile_server_port** are the Mobile Server host and port respectively.

Note: The HTTPS protocol is required, so you would need to provide your HTTPS port.

- app_virtual_path** is the virtual path for your application. It is the same virtual path that you specify when you publish your application using Packaging Wizard.
- app_name.ipa** is the name of the application archive that you are going to save.

Note: If not correctly specified at this time, you can make adjustments to the application URL in the resulting .plist manifest file.

- Title is the name of your application.

Note: To correctly publish your application to the Mobile Server, the name of your application must be the last part of your application bundle id. For example, if your application bundle id is **com.oracle.TransportDemo**, your application name must be **TransportDemo**.

- You can enter additional information in the fields below "Title", though it is optional.
- Click "Save". You will have 2 files with extension .ipa and .plist (for example, TransportDemo.ipa and TransportDemo.plist). The first file is the application archive. The second file is the application manifest.
- Use Packaging Wizard as described in Chapter 5 "Using the Packaging Wizard" of the *Mobile Server Developer's Guide* to publish the application to Mobile Server.

Note:

- For iOS applications, you do not enter application name in Packaging Wizard. Instead you will need to choose your .plist manifest file from which the Packaging Wizard will retrieve the application bundle id and derive the application name accordingly as the last part of the bundle id (see Section 5.1.2 "Specifying New Application Definition Details" of the *Mobile Server Developer's Guide*).
 - The Virtual Path you specify in Packaging Wizard must be the same as the **app_virtual_path** you specified in the Application URL above.
 - You must upload two files as part of your application (see Section 5.1.3 "Listing Application Files" of the *Mobile Server Developer's Guide*): .ipa archive and .plist application manifest.
 - Once application is published, you will find your application files under directory **MOBILE_HOME\Mobile\Server\admin\repository\app_virtual_path**, where **app_virtual_path** is the virtual path you specified above. You can open .plist manifest file in text editor and make adjustments to the application URL if needed.
 - Once published, use Mobile Manager to grant application access to users and groups (see [Section 3.6.1, "Granting Application Access to Users and Groups"](#)).
-
-

11.1.4.3 Application Security Considerations

Access to application archives hosted on your Mobile Server is not restricted by authentication. Anyone who can access your Mobile Server can download application .ipa archive by using the application URL you specified for Enterprise distribution (see [Section 11.1.4.2, "Preparing and Publishing iOS Application"](#)). The reason for this is that iOS application installation process does not currently use client certificate authentication.

This means that you must not store unencrypted sensitive information within your application bundle. Any sensitive information must only be available to an

authenticated user. One way to do this would be to provide a login screen for the user at the application startup.

11.2 Server and Security Configuration for iOS Mobile Device Management

Managing iOS devices with Mobile Server using MDM protocol requires additional configuration. For better understanding of the requirements and the configuration steps, see [Section 11.1, "iOS Device Management with MDM"](#).

The following are additional security features of iOS MDM:

- All the communication between the Mobile Server and iOS devices has to be done over SSL (HTTPS) protocol. Also, self-signed SSL certificates cannot be used for MDM.
- MDM uses client certificate authentication to authenticate iOS devices when they connect to the Mobile Server. Each device needs to be identified by its Identity Certificate. In addition, Over The Air (OTA) device enrollment requires to issue additional temporary certificates. This requires creating and configuring Certificate Authority.
- MDM uses Apple Push Notification Service (APNS) to notify devices that new commands are available. APNS requires Mobile Server to have a valid Push Notification Certificate so that it can authenticate itself to the Apple's APNS Service.

The following sections describe the necessary configuration steps to setup MDM with Mobile Server:

- [Section 11.2.1, "Creating and Configuring Certificate Authority"](#)
- [Section 11.2.2, "SSL Configuration for MDM"](#)
- [Section 11.2.3, "Obtaining and Importing APNS Certificate"](#)

11.2.1 Creating and Configuring Certificate Authority

Mobile Server needs to use Certificate Authority for the following reasons:

- iOS devices enroll in MDM using Over the Air (OTA) profile delivery and configuration. For information on OTA process, see: <https://developer.apple.com/library/ios/documentation/networkinginternet/conceptual/iphoneotaconfiguration/Introduction/Introduction.html>

During the enrollment process a configuration profile with MDM payload is delivered to the device.

Note: This process requires a certificate to be issued for the device. Mobile Server uses similar process as described in the "Over-the-Air Profile Delivery Concepts". See here:

https://developer.apple.com/library/ios/documentation/networkinginternet/conceptual/iphoneotaconfiguration/OTASecurity/OTASecurity.html#//apple_ref/doc/uid/TP40009505-CH3-SW1

The only difference is that currently SCEP protocol is not used to issue device certificates.

Instead, Mobile Server creates and signs device certificates which are then sent to the device over secure connection.

- MDM uses client certificate authentication to verify device identities of iOS devices connecting to Mobile Server. MDM protocol requires to issue an Identity Certificate (certificate bundled together with its private key) for each device.

The Identity Certificate must be unique for each device. During SSL handshake, the device presents its Identity Certificate to the application server. In addition, once the device is authenticated over SSL, Mobile Server reads the certificate and verifies its identity (in fact, Identity Certificates are issued with the Subject Common Name being the device's UDID).

- If you use Mobile Server only to manage iOS devices, you can sign your server's SSL certificate with the CA you created instead of buying SSL certificate from trusted Certificate Authority. The CA that you have will be deployed to each iOS device during enrollment process, which means that the managed iOS devices will trust your server's SSL certificate. For more information, see [Section 11.2.2, "SSL Configuration for MDM"](#).

You do not need to worry about managing iOS device's MDM certificates as Mobile Server takes care of it. The only requirement is that you create or obtain a CA and import it to the location where Mobile Server can find it. This can be done via one of the following:

- Create your own private CA. This will probably be the most common approach. For more information, see [Section 11.2.1.1, "Creating Private CA"](#).
- If you already possess an Enterprise-level Certificate Authority, you can also use it for MDM (an example would be Windows Active Directory CA).

The following sections describe the necessary steps:

- [Section 11.2.1.1, "Creating Private CA"](#)
- [Section 11.2.1.2, "Importing the CA into Mobile Server MDM Keystore"](#)
- [Section 11.2.1.3, "Importing the CA into the Application Server Trust Store"](#)
- [Section 11.2.1.4, "Important Considerations"](#)

11.2.1.1 Creating Private CA

The private certificate authority is a self-signed certificate that is used to sign other certificates. The easiest way to create it is with using **openssl** command line tool. If you are on Linux or Mac, **openssl** must be available. For Windows, you can download **openssl** from here:

<http://slproweb.com/products/Win32OpenSSL.html>

You can find more information on **openssl** from: <https://www.openssl.org>

To create your private CA, do the following:

1. Create private key file:

```
openssl genrsa -out myCA.key 2048
```

This generates 2048 bit private key.

2. Create the CA self-signed certificate:

```
openssl req -x509 -new -key myCA.key -out myCA.cer -days 1000
```

Note: openssl will ask you for some information. Enter as required.

Here is an example:

```
>openssl req -x509 -new -key myCA.key -out myCA.cer -days 1000
Loading 'screen' into random state - done
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
```

```
Country Name (2 letter code) [US]:US
State or Province Name (full name) [Some-State]:California
Locality Name (eg, city) []:Redwood Shores
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Oracle
Organizational Unit Name (eg, section) []:Database Mobile Server
Common Name (e.g. server FQDN or YOUR name) []:My Custom CA
Email Address []:
```

Alternatively, you can skip this by specifying only Subject Common Name:

```
openssl req -x509 -new -key myCA.key -out myCA.cer -days 1000 -subj
/CN="My Custom CA"
```

This will create self-signed CA certificate valid for 1000 days.

3. Package the certificate and the private key together into PKCS12 format (.p12 extension):

```
openssl pkcs12 -export -out myCA.p12 -in myCA.cer -inkey myCA.key
```

Note: This asks you for the export password, choose non-empty password.

11.2.1.2 Importing the CA into Mobile Server MDM Keystore

For iOS MDM, Mobile Server uses Java Keystore (.jks extension). It is located in the following directory:

```
MOBILE_HOME\Mobile\Server\admin\repository\devmgr\apple\mdm\cert
```

where MOBILE_HOME is your Mobile Server installation directory. The keystore file name is **keystore.jks**

The keystore may contain 2 types of entries:

- Trusted Certificate Entry - just certificate with its public key

- Private Key Entry - certificate (with its public key) together with the corresponding private key

The keystore is protected by password. In addition, each Private Key Entry is also protected by its own password.

Note: For Mobile Server MDM Keystore, the password is the same as your Mobile Server Repository password (the database password for your Mobile Server Repository schema). In addition, the same Mobile Server Repository password is used for every Private Key Entry in the MDM Keystore.

You can use Java keytool utility to access and modify Java keystore. For more information on keytool, see <http://docs.oracle.com/javase/6/docs/technotes/tools/solaris/keytool.html>. Keytool should be available wherever JDK is installed.

Note: You must import the CA as a Private Key Entry to the MDM Keystore because its private key will be used to sign MDM certificates.

Following the previous example:

```
cd %MOBILE_  
HOME%\Mobile\Server\admin\repository\devmgr\apple\mdm\cert  
keytool -importkeystore -srckeystore myCA.p12 -srcstoretype pkcs12 -destkeystore  
keystore.jks -sralias 1 -destalias oracle.dms.mdm.ca
```

Keytool asks you for source keystore and destination keystore passwords. For source keystore, enter the password that you chose when creating myCA.p12 (openssl pkcs12...). For destination keystore, remember to use Mobile Server Repository password.

Note: The source alias is specified as 1. When .p12 file is created, most likely alias name 1 is chosen for the private key entry. If you get an error from the previous command, you can always list the contents of .p12 file:

```
keytool -v -list -keystore myCA.p12
```

and find the source alias name.

Note: You will also see that the destination alias is specified as "oracle.dms.mdm.ca". It is important that you enter destination alias exactly as specified above. Mobile Server uses this alias name to find the CA entry.

11.2.1.3 Importing the CA into the Application Server Trust Store

When iOS device connects to Mobile Server, it will first be authenticated by your Application Server's SSL layer. Application Server will do that by examining the device's Identity Certificate which is issued by the CA described above. For application server to trust the device's identity certificate, you need to import the CA

into the application server's trust store. The procedure differs depending on which application server you are using. See the sections below:

- [Section 11.2.1.3.1, "Glassfish"](#)
- [Section 11.2.1.3.2, "WebLogic"](#)

11.2.1.3.1 Glassfish For Glassfish, the trust store is located in your domain subdirectory:

GLASSFISH_HOME\domains\DMS_Domain\config

where GLASSFISH_HOME is the installation directory of the Glassfish Server and DMS_Domain is the domain under which you are running Mobile Server. In this directory you will see 2 Java keystore files:

- keystore.jks
- cacerts.jks

You must import your CA certificate into the trust store cacerts.jks:

keytool -import -keystore cacerts.jks -alias myCA -file myCA.cer

Note: We only need to import the CA as trusted certificate, not the private key entry, so we can use the certificate file **myCA.cer** that we generated previously. Also, the alias name does not matter here so you can choose any alias (that is not already used in cacerts.jks).

You are prompted for the keystore password (the default password for Glassfish keystores is "changeit") and also asked whether to trust the certificate, answer "yes" to this.

11.2.1.3.2 WebLogic

For WebLogic, you can configure the location of your keystore (also called Identity Keystore) and your trust store (also called Trust Keystore). For information on SSL and security configuration, see

http://docs.oracle.com/cd/E13222_01/wls/docs81/secmanage/ssl.html

In short, the process is as below:

- Choose the directory location and file names for your keystore and your trust store. You can choose the same file to use for both of them. For our example, we assume **mykeystore.jks** and **mytruststore.jks**.
- Import your CA certificate in your trust store (same as for Glassfish):

keytool -import -keystore mytruststore.jks -alias myCA -file myCA.cer

Note: This command creates the trust store (**mytruststore.jks**) if it does not exist. Enter password and choose to trust the certificate when prompted.

- Login to the WebLogic Administration Console. It should be at URL: http://myhost:my_admin_port/console
- Navigate to "Servers" (under "Environment" node), choose your server. You will be looking at "Configuration" tab. Ensure "SSL Listen Port" checkbox is selected.

- Click on "Keystores" tab. If you have just installed the server, your setting will be "Demo Identity and Demo Trust". Click "Change" button and from the drop-down list on the next screen choose "Custom Identity" and "Custom Trust". Click "Save".
- Enter path to the keystore file, keystore password, and keystore type (for each Identity Keystore and Trust Keystore), which for our example must be "JKS". Click "Save".
- Switch to "SSL" tab. Ensure that "Identity and Trust Locations" are set to "Keystores" (if it is not, change and save it).

11.2.1.4 Important Considerations

See the sections below:

- [Section 11.2.1.4.1, "Changing Repository Password"](#)
- [Section 11.2.1.4.2, "Dealing with Certificate Expiration"](#)

11.2.1.4.1 Changing Repository Password

Changing repository password is described in [Section 9.2.1, "Modifying Repository Password"](#). After you change the repository password, you must also change the password of the MDM keystore (see [Section 11.2.1.2, "Importing the CA into Mobile Server MDM Keystore"](#)), because the password of the MDM keystore must match your repository password. You can do it with this command:

```
cd %MOBILE_
HOME%\Mobile\Server\admin\repository\devmgr\apple\mdm\cert
keytool -storepasswd -keystore keystore.jks
```

You are prompted for old and new repository passwords.

Ensure to restart the server (you can do the command above between steps 4 and 5 in [Section 9.2.1, "Modifying Repository Password"](#)).

11.2.1.4.2 Dealing with Certificate Expiration

Mobile Server issues device certificates for the same duration as the duration of your CA. When your CA certificate expires, you must create new CA according to the procedures outlined above and re-import it into MDM keystore and your Application Server truststore. This can be done the same way as described above, except that before importing the new entry into keystore, you must delete the old entry that has the same alias:

```
keytool -delete -alias alias_name -keystore keystore_name
```

where, **alias_name** and **keystore_name** are the names of the alias and the keystore file.

We do not have a way to automatically provision new certificates to all managed devices, so when CA certificate expires, all managed devices will have to re-provision new CA certificate from Mobile Server setup page and re-enroll into MDM. It is recommended to use a long duration for your CA certificate.

If you are concerned about security, you can always increase the bit length of your private key (see **openssl genrsa** command in [Section 11.2.1.1, "Creating Private CA"](#)). The device certificates will be issued with the same bit length (this will have a tradeoff for RSA performance). It is advisable to use 4096 bit keys for certificates with long duration.

11.2.2 SSL Configuration for MDM

For MDM to function, run Mobile Server in SSL mode. For information on SSL setup process, see [Section 9.4, "Configuring for Secure Socket Layer \(SSL\) Communication"](#). However, the self-signed server SSL certificate with which most application servers come by default will not work for MDM because iOS MDM process will not trust self-signed SSL certificate. There are two options:

- Obtain server certificate from trusted Certificate Authority. This process is briefly described in [Section 9.4.1, "Creating an SSL Certificate"](#).
- Use the private CA that you created (see [Section 11.2.1, "Creating and Configuring Certificate Authority"](#)) to sign your server's SSL certificate. Since all managed iOS devices will have to be provisioned with your private CA certificate and your server's SSL certificate is signed with the same CA, iOS devices will trust your server's SSL certificate. The procedure for this is described in the following section: [Section 11.2.2.1, "Creating SSL Certificate Signed by Private CA"](#).

11.2.2.1 Creating SSL Certificate Signed by Private CA

You can use SSL certificate signed by your private (or enterprise-level) Certificate Authority if any of the following applies:

- You plan to use Mobile Server only to manage iOS devices.
- You plan to use Mobile server to manage other devices as well, but you do not need SSL functionality for other platforms. In this case, other devices can access Mobile Server through regular HTTP.
- You plan to use SSL for other devices, but can provision the CA certificate to those devices.
- You plan to use SSL for other devices, but also set `DISABLE_SSL_CHECK` parameter to `TRUE` (see [Section 9.4.1, "Creating an SSL Certificate"](#))

If none of the above applies, you must obtain your SSL certificate from the trusted certificate authority.

To create SSL certificate signed by your CA, you can use "`openssl`" (see [Section 11.2.1, "Creating and Configuring Certificate Authority"](#)):

```
openssl genrsa -out svrcert.key 2048
```

```
openssl req -new -out svrcert.csr -key svrcert.key
```

```
Loading 'screen' into random state - done
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:US
State or Province Name (full name) [Some-State]:California
Locality Name (eg, city) []:Redwood Shores
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Oracle
Organizational Unit Name (eg, section) []:DMS
Common Name (e.g. server FQDN or YOUR name) []:pc1.oracle.com
Email Address []:
```

```
Please enter the following 'extra' attributes
to be sent with your certificate request
```

A challenge password []:

An optional company name []:

```
openssl x509 -req -in svrcert.csr -out svrcert.cer -CAkey myCA.key -CA myCA.cer  
-days 1000 -CAcreateserial -CAserial serial
```

```
openssl pkcs12 -export -out svrcert.p12 -in svrcert.cer -inkey svrcert.key
```

- The "**openssl req**" command creates a certificate service request (CSR) to be signed by your CA. It asks you for some information. You can enter most as you think required, except the following:

Note: The Common Name of the certificate must match your server's host name (which can be fully qualified domain name) or IP address. This is required for SSL to work. You should use the same hostname (or ip address) that iOS devices will use to enroll in MDM.

- The "**openssl x509**" command creates a certificate based on the CSR you created signed by your CA. It uses files **myCA.key** and **myCA.cer** created previously (see [Section 11.2.1.1, "Creating Private CA"](#)). The options **-CAcreateserial** and **-CAserial** facilitate creating serial numbers for the certificates in case you need to create multiple certificates with successive serial numbers.
- The "**openssl pkcs12**" command packages your certificate and private key into PKCS12 bundle so that it can be easily imported into Java keystore.

11.2.2.2 Importing SSL Certificate into Application Server's Keystore

Once you have your SSL certificate signed either by trusted CA or your private CA, you must import it into your Application Server's keystore. It must be imported as Private Key Entry (containing both certificate and the private key). The procedure differs depending on which application server you are using. See the sections below:

- [Section 11.2.2.2.1, "Glassfish"](#)
- [Section 11.2.2.2.2, "WebLogic"](#)

Note: For more information on how application servers are configured for MDM, see [Appendix D.3, "Application Server Configuration for MDM."](#)

11.2.2.2.1 Glassfish For Glassfish, the keystore is located in your domain subdirectory:

```
GLASSFISH_HOME\domains\DMS_Domain\config
```

(see [Section 11.2.1.3.1, "Glassfish"](#)) and is named **keystore.jks**. The easiest way is to import it at the same alias as the default self-signed certificate created by Glassfish, thus replacing default entry.

For this you must delete the default entry first. The alias name of the default entry should be "**s1as**".

You can also login to the Glassfish Administration Console, go to SSL settings and check the certificate alias name. It will also give you an option to change it if you wish to do so. Continuing from the example above, you can issue the following commands:

```
cd GLASSFISH_HOME\domains\DMS_Domain\config
```

```
keytool -delete -alias s1as -keystore keystore.jks
```

```
keytool -importkeystore -srckeystore svrcert.p12 -srcstoretype pkcs12 -destkeystore  
keystore.jks -srcaalias 1 -destalias s1as -trustcacerts
```

The last option (**-trustcacerts**) is used to look for certificate chain in file **cacerts.jks** since this is where you imported your CA certificate previously (see [Section 11.2.1.3.1, "Glassfish"](#)). This allows the keytool to trust your certificate so that it can be successfully imported.

You are prompted for destination and source passwords. For source password, use the one you entered during "**openssl pkcs12**" command. For destination password, use the Glassfish keystore password, which by default is "**changeit**".

Once the configuration is done, restart Mobile Server and check that you can connect to it via SSL. Before that, you can import your CA certificate into the browser. If you do that, or if your SSL certificate was obtained from trusted CA, you must be able to connect without seeing "untrusted" warning from the browser.

11.2.2.2 WebLogic For WebLogic Server, you can configure the location of your keystore (also called Identity Keystore) and your trust store (also called Trust Keystore). For information on importing your CA into the Trust Keystore, see [Section 11.2.1.3.2, "WebLogic"](#). Follow the steps below:

- Choose the directory location and file names for your keystore and your trust store (see [Section 11.2.1.3.2, "WebLogic"](#)). For our example, we assume **mykeystore.jks** and **mytruststore.jks**.
- Import your SSL certificate/private key into your keystore in the same way as it is done for Glassfish (see [Section 11.2.2.1, "Glassfish"](#)):

```
keytool -importkeystore -srckeystore svrcert.p12 -srcstoretype pkcs12  
-destkeystore mykeystore.jks -srcaalias 1 -destalias mysslalias -trustcacerts
```

This command creates the keystore (**mykeystore.jks**) if it does not exist. At this point you can choose your destination alias name (**mysslalias** in the example).

- Login to the WebLogic Administration Console. It must be at URL:
`http://myhost:my_admin_port/console`
- Navigate to "Servers" (under "Environment" node), choose your server. You will be looking at "Configuration" tab. Ensure "SSL Listen Port" checkbox is selected.
- Click on "Keystores" tab. If you just installed the server, your setting will be "Demo Identity and Demo Trust". Click "Change" button and from the drop-down list on the next screen choose "Custom Identity and Custom Trust". Click "Save".
- Enter path to the keystore file, keystore password, and keystore type (for each Identity Keystore and Trust Keystore), which for us must be "JKS". Click "Save".
- Switch to "SSL" tab. Ensure that "Identity and Trust Locations" are set to "Keystores" (if it is not, change and save it).
- Enter the alias under which you imported your SSL private key (**mysslalias** in the example above), on the same page, in the "Identity" panel (under "Identity and Trust Locations"). Also enter private key passphrase which must be your identity keystore (**mykeystore.jks**) password. Click "Save".

Once the configuration is done, restart Mobile Server and check that you can connect to it via SSL. Before that, you can import your CA certificate into the browser. If you do that, or if your SSL certificate was obtained from trusted CA, you must be able to connect without seeing "untrusted" warning from the browser.

11.2.3 Obtaining and Importing APNS Certificate

MDM protocol uses Apple Push Notification Service to notify the managed devices when new commands are available for them. For more information on APNS, see: <https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html>

For information on how Apple Push Notifications are used in MDM, see, [Section 11.1, "iOS Device Management with MDM"](#).

Mobile Server must have APNS certificate so that APNS service can authenticate it. Unlike APNS certificates for iOS applications, APNS certificate for MDM is not tied to a particular application id, and a slightly different procedure is used to obtain it.

Apple recognizes the following two roles:

- MDM Vendor
- MDM Customer

MDM Vendor is a company providing MDM solution (for example, Oracle). MDM Customer is a company that buys MDM solution to manage iOS devices. The following steps outlines the procedure to obtain APNS certificate:

- Vendor obtains MDM CSR Signing Certificate.
- Customer creates a private key and Certificate Signing Request (CSR) for the APNS certificate and submits it to the Vendor.
- Vendor signs the CSR submitted with their MDM CSR Signing Certificate and returns signed CSR to the Customer.
- Customer submits the signed CSR to Apple Push Certificate Portal and the portal issues APNS certificate which the customer can download.
- Customer imports APNS certificate and its private key into MDM keystore.

For more information, see the sections below:

- [Section 11.2.3.1, "Obtain MDM CSR Signing Certificate"](#)
- [Section 11.2.3.2, "Create APNS CSR"](#)
- [Section 11.2.3.3, "Sign CSR with MDM CSR Signing Certificate"](#)
- [Section 11.2.3.4, "Submit CSR and Obtain APNS Certificate"](#)
- [Section 11.2.3.5, "Import APNS Certificate and Private Key into Mobile Server MDM Keystore"](#)
- [Section 11.2.3.6, "Renewing MDM Certificates"](#)
- [Section 11.2.3.7, "Important Considerations"](#)

Note: A Customer having an iOS Developer Enterprise Account can easily become a MDM Vendor by doing one of the following:

- Contact Apple
(<https://idmsa.apple.com/IDMSWebAuth/login.html?path=%2F%2Fcontact%2Fsubmit.php&appIdKey=891bd3417a7776362562d2197f89480a8547b108fd934911bcbea0110d07f757>) and ask to be assigned MDM Vendor role
- Talk to your team agent to be assigned appropriate privileges to create MDM Certificates

Since most customers must have Enterprise account anyway to develop and distribute Enterprise applications, they can also assume the Vendor role. This is most expedient option since this allows you to obtain APNS certificate without any external dependencies.

Thus, in this guide we will describe the steps for both Vendors and Customers.

If you do not wish to assume the Vendor role, contact Oracle Support for the Vendor-specific steps.

11.2.3.1 Obtain MDM CSR Signing Certificate

Follow the steps below:

- Go to developer portal at <https://developer.apple.com/>, click on "Member Center" (top right) and sign in with your Apple ID.
- Click on "Certificates, Identities & Profiles".
- Navigate to "Certificates" and click "+" (top right of the table on the right).
- Select "MDM CSR" (it will be under "Production"), on the page that appears and click "Continue" button.
- Generate CSR first to get the Vendor certificate. In the following screen you will see instructions on how to generate CSR using Mac's Keychain utility. However, you can also generate CSR using `openssl` (see [Section 11.2.2.1, "Creating SSL Certificate Signed by Private CA"](#)):
 - `openssl genrsa -out vendor.key 2048`
 - `openssl req -new -out vendor.csr -key vendor.key`

Ensure that you enter your name (CN) and your email address when prompted by `openssl`. If instead you choose using Keychain utility, you need to export your private key from it as PKCS12 bundle (ext .p12 or .pfx).

On the portal, click "Continue" button.

- Select your generated CSR file (on the next screen) to upload.
- Download your MDM CSR Signing Certificate (that is generated) from the next screen.
- See the row with your newly generated certificate (back at the "Certificates" table on the portal under "Production"). You can re-download it from the portal later if you wish. The "Type" column should indicate "MDM CSR".

11.2.3.2 Create APNS CSR

To generate APNS private key and CSR you can use **openssl**, see [Section 11.2.2.1, "Creating SSL Certificate Signed by Private CA"](#):

- **openssl genrsa -out push.key 2048**
- **openssl req -new -out push.csr -key push.key**

You are asked for some information by "**openssl req**" command. Enter the information for your organization (enter at least your name (subject CN) and email address). Alternatively, you can generate private key and CSR by Apple Keychain Utility on Mac.

Save the CSR instead of discarding it because it can be reused when your APNS certificate needs to be renewed.

11.2.3.3 Sign CSR with MDM CSR Signing Certificate

If you do not want/cannot assume the Vendor role, submit the CSR you created to Oracle Support. Otherwise, follow the steps below. Although there is no official Apple tool for MDM CSR signing, Mobile Server provides an MDM CSR signing utility. The utility is located in file **applemdm.jar**. You can find **applemdm.jar** in your Mobile Server subdirectory:

MOBILE_HOME\mobile\server\admin\repository\mobile\lib.

To use the MDM CSR signing utility you must get the following:

- Vendor's CSR Signing Certificate.

It can be either in PEM format (text, base-64 encoding) with extension **.pem** or in binary DER format, with extensions **.cer** or **.der**. Refer to [Section 11.2.3.1, "Obtain MDM CSR Signing Certificate"](#) on how to obtain it.
- Vendor's CSR Signing private key.

The private key has to be in PKCS12 format in a file with extension **.p12** or **.pfx**. If you exported the private key from the Mac Keychain utility, it must already be in PKCS12 format. On the other hand, if you used **openssl** to generate your vendor's private key and CSR as in [Section 11.2.3.1, "Obtain MDM CSR Signing Certificate,"](#) you can convert the key file generated by **openssl** to PKCS12 format:

```
openssl pkcs12 -export -nocerts -inkey vendor.key -out vendor.p12
```

-nocerts option tells **openssl** to generate **vendor.p12** containing only private key and no certificates. You will be prompted for the password to protect **.p12** file.
- Customer's CSR

See [Section 11.2.3.2, "Create APNS CSR."](#)
- Apple Root Certificate

Download it here:
<http://www.apple.com/appleca/AppleIncRootCertificate.cer>

The file name should be **AppleIncRootCertificate.cer**.
- Apple WWDR Intermediate Certificate

Download it here:
<http://developer.apple.com/certificationauthority/AppleWWDRCA.cer>

The file name should be **AppleWWDRCA.cer**.

Run MDM CSR Signing utility as follows:


```
java -cp applemdm.jar oracle.opensync.apple.tools.MDMCSRSign push.csr
vendor.p12 vendor_pk_password vendor.pem AppleWWDRCA.cer
AppleIncRootCertificate.cer > push.signedcsr
```

where:

- **push.csr** - Customer's APNS CSR
- **vendor.p12 (or vendor.pfx)** - Vendor's CSR Signing private key as PKCS12 bundle.
- **vendor_pk_password** - Password protecting vendor.p12 (or vendor.pfx).
- **vendor.pem (or vendor.cer)** - Vendor's CSR Signing Certificate that was downloaded in section 11.2.3.1.
- The last 2 parameters are Apple's WWDR intermediate and Apple's root certificates.

Note: If their names match the names given above (AppleWWDRCA.cer and AppleIncRootCertificate.cer), these last 2 parameters can be omitted.

- **push.signedcsr** - The result of running the utility, Customer's APNS signed CSR (the utility writes the result to standard output, thus the re-direction).

Save the signed CSR instead of discarding it because it can be reused when your APNS certificate needs to be renewed.

11.2.3.4 Submit CSR and Obtain APNS Certificate

Follow the steps below:

1. Go to Apple Push Certificate Portal.
2. Sign in with your Apple ID.
3. Click on "Create a Certificate" button.
4. Read and accept terms and conditions.
5. Select the signed CSR to upload (for example, **push.signedcsr**)
6. Download APNS certificate (newly generated) from the following screen.

11.2.3.5 Import APNS Certificate and Private Key into Mobile Server MDM Keystore

The last step is to import APNS certificate into Mobile Server MDM keystore. See [Section 11.2.1.2, "Importing the CA into Mobile Server MDM Keystore"](#) for details on the MDM keystore.

You must combine your APNS private key and APNS certificate into PKCS12 bundle.

Note: Mobile Server MDM keystore password is the same as your Mobile Server repository password.

Here are the commands:

```
cd %MOBILE_
HOME%\Mobile\Server\admin\repository\devmgr\apple\mdm\cert
openssl pkcs12 -export -out push.p12 -in push.pem -inkey push.key
```

```
keytool -importkeystore -srkeystore push.p12 -srcstoretype pkcs12 -destkeystore  
keystore.jks -sralias 1 -destalias oracle.dms.mdm.apns
```

where:

- **push.pem** - APNS certificate you downloaded from the portal (see [Section 11.2.3.4, "Submit CSR and Obtain APNS Certificate"](#))
- **push.key** - APNS private key (see [Section 11.2.3.2, "Create APNS CSR"](#))
- **push.p12** - bundle combining APNS certificate with APNS private key
- **oracle.dms.mdm.apns** - destination alias in MDM keystore.

Note: It is important that you enter destination alias exactly as specified above. Mobile Server uses this alias name to find the APNS Certificate entry.

After this is completed, restart your Mobile Server.

11.2.3.6 Renewing MDM Certificates

Both Vendor's CSR Signing Certificate and Customer's APNS Certificate have limited duration (currently 1 year), so they will need to be renewed. Follow the steps below for renewal:

- **For Vendor's CSR Signing Certificate:**

Use the procedure outlined in [Section 11.2.3.1, "Obtain MDM CSR Signing Certificate,"](#) to obtain new certificate. No special renewal process is needed.

- **For Customer's APNS Certificate:**

You must renew the certificate before it expires. If you do not renew it and obtain a new certificate instead, Mobile Server will not be able to push notifications to currently enrolled devices until they re-enroll into MDM. Follow the steps below:

- Go to Apple Push Certificate Portal.
- Sign in with your Apple ID.
At the top, you will see a table named "Certificates for Third-Party Servers".
- Find your APNS certificate in this table and click "Renew" button on the right.
You will be taken to a page to upload your signed CSR.
- Reuse the same signed CSR that you created previously as long as the Vendor's CSR signing certificate used to sign this CSR has not expired. Otherwise, you will need to sign your CSR (see [Section 11.2.3.2, "Create APNS CSR"](#)) again following procedure in [Section 11.2.3.3, "Sign CSR with MDM CSR Signing Certificate."](#)
- Download the new certificate issued (see [Section 11.2.3.4, "Submit CSR and Obtain APNS Certificate"](#)) in the following page.
- Import the new certificate into MDM keystore as described in [Section 11.2.3.5, "Import APNS Certificate and Private Key into Mobile Server MDM Keystore."](#)

Note: You must first delete the old entry from the keystore:

```
keytool -delete -alias oracle.dms.mdm.apns -keystore keystore.jks
```

11.2.3.7 Important Considerations

See the following section:

- [Section 11.2.3.7.1, "Push Notification Topic"](#)

11.2.3.7.1 Push Notification Topic Similar to push notifications sent by iOS Apps, MDM protocol must specify push notification topic. The push notification topic for MDM must start with prefix **com.apple.mgmt**. The topic is located in the APNS certificate's Subject UID field. You can check this with **openssl**:

openssl x509 -noout -text -in push.pem

Certificate:

```

Data:
  Version: 3 (0x2)
  Serial Number:
    20:b6:b9:9e:0b:6f:c2:09
  Signature Algorithm: sha1WithRSAEncryption
  Issuer: C=US, O=Apple Inc., OU=Apple Certification Authority, CN=Apple A
pplication Integration Certification Authority
  Validity
    Not Before: Aug 16 07:33:54 2014 GMT
    Not After : Aug 16 07:33:54 2015 GMT
  Subject: UID=com.apple.mgmt.
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
0, CN=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX, C=US
  Subject Public Key Info:
.....

```

It is useful to check if you get any push notification related errors while running MDM in Mobile Server.

Configuration Files for the Mobile Server

The following sections describe configuration parameters for the mobile server:

- [Section A.1, "Configuration Parameters for the MOBILE.ORA File"](#)
- [Section A.2, "Data Synchronization Requirements in INIT.ORA"](#)

A.1 Configuration Parameters for the MOBILE.ORA File

The mobile server uses the `mobile.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 in the `mobile.ora` file:

- [Section A.1.1, "\[APPLICATIONS\]"](#)
- [Section A.1.2, "\[MOBILE\]"](#)
- [Section A.1.3, "\[FILESYSTEM\]"](#)
- [Section A.1.4, "\[DEBUG\]"](#)
- [Section A.1.5, "\[CONSOLIDATOR\]"](#)
- [Section A.1.6, "\[EXTERNAL_AUTHENTICATION\]"](#)

A.1.1 [APPLICATIONS]

[Table A-1](#) lists the `APPLICATIONS` parameters and their usage definitions.

Table A-1 *APPLICATIONS Parameters*

Parameter	Definition
<code>PACK_HELP</code>	Location for the Packaging Wizard help file. This is only used with MDK, and only for the purpose of Debug / Support.
<code>XMLFILE</code>	Name of the XML file used by the Packaging Wizard to store the application information. This is only used for Debug / Support.

A.1.2 [MOBILE]

The following `MOBILE` parameters control the behavior of the mobile server.

[Table A-2](#) lists `MOBILE` parameters and their usage definitions.

Table A-2 MOBILE 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>/mobile/console/startup
ADMIN_PORT=8080	Admin port for starting the mobile server.
ADMIN_JDBC_URL=<jdbc_url>	JDBC URL that the mobile server uses to connect to the mobile server repository in the back-end Oracle database. For a description of the syntax for the JDBC URL, see Section 2.2, "Connecting to the Back-End Oracle Database" .
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>/mobile/console/startup
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.
INSTALLATION_TYPE	Mobile Server Installation Type, which describes the application server type where the mobile server is running. Do NOT modify, for internal use only.
IP_CONFIG	Set this parameter to designate what type of IP address the client uses. If you are using a dynamic (DHCP) dynamic IP address, set the parameter to DYNAMIC; the default is STATIC for a static IP address. If using DHCP, 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, synchronization may never occur, since it is 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 <code>networkaddress.cache.ttl</code> 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 Oracle Java documentation for more information.
MAX_THREAD_POOL	Limits the number of threads available in the connection pool. If threading problems occur, set this parameter to 0 or 1.
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.

Table A–2 (Cont.) MOBILE Parameters

Parameter	Definition
RESTRICTED_ADMIN_HOSTS=<list of comma separated IP addresses>	<p>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.</p> <p>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.</p> <p>For example, RESTRICTED_ADMIN_HOSTS=144.125.127.150,144.125.127.101</p> <p>Note: Users who have administrator access should not connect through a proxy server.</p>
REVERSE_PROXY=http://<proxyhost>:<port>/mobile	Set this parameter if a Reverse proxy is used with the mobile server. See Section 9.8.1, "Using a Reverse Proxy to Communicate from Internet to Intranet" for details.
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.
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.
THIN_JDBC_URL=<jdbc_url>	<p>This URL is used by applications to connect to the mobile server repository database.</p> <p>For a description of the syntax for the JDBC URL, see Section 2.2, "Connecting to the Back-End Oracle Database".</p>
USE_SYSTEM_CLASSPATH=YES	If set to yes, searches for Java classes in the computer's classpath before searching the mobile server repository.

A.1.3 [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
ROOT_DIR=ORACLE_HOME/MOBILE/SERVER/REPOSITORY	<p>Root Directory. Valid only for OS file system.</p> <p>This directory path format applies to the environment where the mobile server runs on Unix, AIX and Solaris.</p> <p>Replace <i>ORACLE_HOME</i> with your actual Oracle Home.</p>
ROOT_DIR=ORACLE_HOME\MOBILE\SERVER\REPOSITORY	<p>Root directory. Valid only for OS file system.</p> <p>This directory path format applies to the environment where the mobile server runs on Windows NT.</p> <p>Replace <i>ORACLE_HOME</i> with your actual home.</p>

A.1.4 [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_DESTINATION	<p>Trace destinations are CONSOLE and FILE. The administrator can set this parameter to any of these destinations.</p> <p>CONSOLE generates trace output to the console screen.</p> <p>FILE generates trace output to a file. For more information, see TRACE_FILE_NAME, TRACE_FILE_SIZE, and TRACE_FILE_POOL_SIZE.</p>
TRACE_ENABLE	Used to turn the trace feature on (YES) or off (NO). When the Trace feature is off, trace output is not generated.
TRACE_FILE_NAME=trace.log	<p>Used as base name to arrange trace files in sequential order starting from 1 to FILE_TRACE_POOL_SIZE.</p> <p>For example: If you set the following parameters.</p> <pre>TRACE_FILE_NAME=mytrace.log TRACE_FILE_POOL_COUNT=5</pre> <p>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.</p> <p>Sample Value: trace.log</p>
TRACE_FILE_PER_USER=YES	<p>Used to specify an individual trace file pool for every individual user. Applicable only when the File option is the Trace destination.</p> <p>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.</p> <p>If set to NO, all traceable users share the same trace file pool, the actual trace file does not contain any user name.</p> <p>Example: TRACE_FILE_POOL_PER_USER=No</p>
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
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.

Table A-4 (Cont.) DEBUG Parameters

Parameter Name	Definition
TRACE_LEVEL=1	<p>There are three levels of trace messages:</p> <p>1(binary 00000001), Basic Trace: General system information.</p> <p>2 (binary 00000010), Function Trace: Traces the function sequence being called, mostly used by Data Synchronization.</p> <p>4 (binary 00000100), SQL Trace: Traces SQL queries being executed, mostly used by Data Synchronization.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>3 & 1 (Basic) = 1 > 0 3 & 2 (SQL) = 2 > 0 3 & 4 (Function) = 0 = 0</p>
TRACE_REMOTE_HOST	<p>The trace output is generated to the named machine, on which the synchronized trace can be viewed using the wsh -m %TRACE_REMOTE_PORT%.</p> <p>Example: TRACE_REMOTE_HOST=adminhost</p>
TRACE_REMOTE_MACHINE	<p>Machine name where wsh -m is running. The Mobile Server sends the debug output to the machine where wsh -m is running.</p>
TRACE_REMOTE_PORT	<p>Trace output is generated to the named port on the %TRACE_REMOTE_MACHINE% on which the trace output can be viewed using wsh -m TRACE_REMOTE_PORT%.</p>
TRACE_USERS	<p>List of valid user names. The user trace and system trace information which is listed is generated as trace output.</p> <p>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.</p> <p>Note: As the administrator, you must not use the TRACE_NO_USER value as the user name.</p> <p>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.</p>

A.1.5 [CONSOLIDATOR]

The CONSOLIDATOR parameters control the behavior of Data Synchronization. The values that are listed in the following table are default values.

- [Section A.1.5.1, "Data Synchronization Parameters"](#)
- [Section A.1.5.2, "Data Synchronization Tracing and Logging"](#)

A.1.5.1 Data Synchronization Parameters

Table A-5 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	<p>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.</p> <p>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.</p>
CONNECTION_TIMEOUT	Specifies the JDBC connection timeout in minutes for the synchronization session. Default value is 120 minutes.
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 consistently 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	<p>Configure whether to validate the database connection before releasing it back to the connection pool. By default, this value is YES.</p> <p>If the examination determines that the connection is not valid, then it is destroyed instead of being returned to the connection pool.</p>
CONNECTION_POOL=YES	Enables pooling of database connections if set to YES.

Table A-5 (Cont.) Consolidator Parameters

Parameter Name	Definition
DO_APPLY_BFR_COMPOSE	<p>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.</p> <p>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.</p>
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 <i>Oracle Database SQL Reference</i> 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 <i>Oracle Database SQL Reference</i> 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 MOBILE section of the mobile.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.
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	<p>Control the magic number checking of publication items. If enabled, and there is a mismatch between the server and the client magic numbers, 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, the magic check is enabled only for shared publication items.</p> <p>A shared publication item has the following characteristics:</p> <ul style="list-style-type: none"> ■ Read only ■ The publication item query either has no parameters or all users share the same parameter values.
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 <i>Oracle Database SQL Reference</i> for more information on these properties.

Table A-5 (Cont.) Consolidator Parameters

Parameter Name	Definition
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 <i>Oracle Database SQL Reference</i> 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_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 concurrent threads spawned within the MGP process. As a rule, do not set this higher than 1.5 times the number of CPUs on the database machine. The default is 3.
MAX_CONNECTIONS	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.
MAX_U_COUNT	The MAX_U_COUNT parameter controls the number of SQL statements that are executed together in a SQL batch statement while performing the map cleanup. The default value for the MAX_U_COUNT parameter is 256. However, if the value is 256 during the map cleanup, then a maximum of 256 SQL statements can be executed together in a batch. Modify this parameter and restart the mobile server to enable a larger batch of SQL statements to be processed during map cleanup. You may want to modify the MAX_U_COUNT parameter before the synchronization starts. See Section 2.7.1.10, "BeforeSyncMapCleanup" in the <i>Oracle Database Mobile Server Developer's Guide</i> for more information.
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.
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.

Table A–5 (Cont.) Consolidator Parameters

Parameter Name	Definition
RESUME_FILE	<p>This is a server-side parameter that defines the filename of the resume buffer for the client users. 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.</p> <p>For more information on how to use RESUME_FILE and RESUME_FILE_SIZE, see Section 5.7, "Resuming an Interrupted Synchronization".</p>
RESUME_FILE_SIZE	Set the maximum size of the resume file in MegaBytes (MB).
RESUME_MAXACTIVE	<p>The RESUME_MAXACTIVE parameter controls the maximum number of connections that 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.</p> <p>You can enable maximum concurrent clients by setting RESUME_MAX_WAIT and RESUME_MAXACTIVE. This limits the maximum number of concurrently synchronizing clients to RESUME_MAXACTIVE; additional incoming clients wait RESUME_MAX_WAIT before timing out. To eliminate the resume feature, set RESUME_TIMEOUT to 0.</p>
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_MAX_WAIT	The RESUME_MAX_WAIT parameter specifies the number of MINUTES a new client waits for a connection if the RESUME_MAXACTIVE threshold has been reached. Default is 30 minutes.
RESUME_CLIENT_TIMEOUT	The RESUME_CLIENT_TIMEOUT parameter specifies the number of SECONDS the client will try to resume before it gives up and returns an error.
RESUME_TIMEOUT	The RESUME_TIMEOUT parameter indicates time in MINUTES the server will 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	<p>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.</p> <p>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.</p>

Table A-5 (Cont.) Consolidator Parameters

Parameter Name	Definition
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.
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 Database Mobile Server 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.1.5.2 Data Synchronoization Tracing and Logging

Data Synchronization uses a log engine that supports the following parameters for logging:

- GLOBALLogger
- SYNCLogger
- MGPLLogger
- 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.

```
<X>Logger=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-6 lists the parameters for each logger.

Table A-6 Acceptable Parameter Values

Parameter	Description
TRACE_LEVEL	<p>Trace Level parameter can be set to the following trace message levels:</p> <p>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.</p> <p>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.</p> <p>NORMAL: This option logs normal messages that the user must be informed with and messages at the Mandatory and Warning level.</p> <p>INFO: This option logs information messages and messages at the Mandatory, Warning, and Normal levels.</p> <p>Examples:</p> <ul style="list-style-type: none"> ■ 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 MGPLLogger is set to the TRACE_TYPE=GENERAL and TRACE_LEVEL=INFO. <p>CONFIG: This option logs configuration messages and messages at the Mandatory, Warning, Normal, and Info levels. For example, JDBC driver version.</p> <p>FINEST: The finest level. This level is used for developers only.</p> <p>ALL: This option logs all messages according to the other settings such as Trace Type and Users.</p>

Table A–6 (Cont.) Acceptable Parameter Values

Parameter	Description
TRACE_TYPE	<p>SQL: This option logs SQL-related messages only. For example, SQL statements. Note: This option is not trace level sensitive.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>GENERAL: This option logs messages that do not belong to any of the above listed trace types. Note: This type is trace level sensitive.</p> <p>ALL: This option generates logs of all trace types.</p>
TRACE_DESTINATION	<p>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.</p> <p>Sample Value: TRACE_DESTINATION=TEXTFILE</p>
TRACE_FILE_POOL_SIZE=2	<p>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.</p>
TRACE_FILE_SIZE=1	<p>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.</p>
TRACE_USERS	<p>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.</p>

The new log engine does not support the parameters that have been used in the old log engine. They are:

- TRACE_REMOTE_PORT
- TRACE_REMOTE_MACHINE
- TRACE_REMOTE_HOST

A.1.6 [EXTERNAL_AUTHENTICATION]

The parameters in [Table A-7](#) are used for configuring external authentication.

Table A-7 EXTERNAL_AUTHENTICATION

Parameter	Description
CLASS=<sampleAuthenticator>	The external Authenticator JAVA class name.
EXPIRATION = <numberOfSeconds>	The mobile server caches the user instantiated through the external authenticator for a period of time in order to improve efficiency. Provide the number of sections for this period. The default expiration time for the cached user object is 30 minutes (or 1800 seconds).

A.2 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 A.2.1, "Relationships Between Relevant Parameters"](#)
- [Section A.2.2, "Values for Processes and DML Locks"](#)

A.2.1 Relationships Between Relevant Parameters

You should set the following parameters in the file `init.ora` as given below:

[Table A-8](#) lists parameters that must be set in the file `init.ora`:

Table A-8 *init.ora* Parameter Settings

Parameter Name	Definition
PROCESSES	Default value: 59 to 200.
SESSIONS	Default value: Derived: $1.1 * \text{PROCESSES} + 5$
TRANSACTIONS	Default value: Derived: $(1.1 * \text{SESSIONS})$
DML_LOCKS	Default Value: Derived: $(4 * \text{TRANSACTIONS})$

A.2.2 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:

$\text{DML_LOCKS} = (\text{Number of changed publications}) * (\text{Maximum number of concurrent clients})$

During the first and second sync, all publication map tables are changed for each client. So, the required DML locks are:

$DML_LOCKS = (\text{Number of publications}) * (\text{Maximum number of concurrent clients})$

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.

Write Scripts for the Mobile Server With the WSH Tool

You can use the scripting language saved in an INI file and used by the WSH tool to perform batch processing tasks on the mobile server that are performed frequently by the administrator.

The following sections describe the scripting language for the mobile server:

- [Section B.1, "Description of Syntax for WSH Batch Scripts"](#)
- [Section B.2, "Running a Script INI File With the WSH Tool"](#)
- [Section B.3, "Examples of Batch Script Files for WSH"](#)

B.1 Description of Syntax for WSH Batch Scripts

The following sections describe the parameters and syntax available in the scripting language:

- [Section B.1.1, "Creating a User"](#)
- [Section B.1.2, "Creating a Group"](#)
- [Section B.1.3, "Adding Users to a Group"](#)
- [Section B.1.4, "Removing Users from a Group"](#)
- [Section B.1.5, "Creating Access Privileges"](#)
- [Section B.1.6, "Granting Access"](#)
- [Section B.1.7, "Revoking Access"](#)
- [Section B.1.8, "Creating Registries"](#)
- [Section B.1.9, "Creating Snapshot Variables"](#)
- [Section B.1.10, "Deleting a User"](#)
- [Section B.1.11, "Deleting a Group"](#)
- [Section B.1.12, "Deleting Access Privileges"](#)
- [Section B.1.13, "Deleting a Registry"](#)
- [Section B.1.14, "Deleting Snapshot Variables"](#)

B.1.1 Creating a User

Using the following syntax, you can create users.

```
[USER]
NAME=<User Name>
PASSWORD=<User Password>
EXTERNALUSER=<True or False>
ENCRYPTED=<True or False; True if the password is encrypted, False if not>
FULLNAME=<User Full Name>
PRIVILEGE=<User privilege level as A, O, U, or null>
```

B.1.1.1 EXTERNALUSER Parameter

By default, this value is false for a user with password that is authenticated by the mobile server. If you are creating an external user that will be authenticated by an external authenticator class, set to True. If true, you do not provide a password in the INI script file with PASSWORD as the user is authenticated by the external authenticator. See Chapter 7, "Customizing Oracle Database Mobile Server Security" in the *Oracle Database Mobile Server Developer's Guide* for information on the external authenticator.

B.1.1.2 PRIVILEGE Parameter

There are four options for setting the PRIVILEGE value for users. They are:

- A - Administrator
- O - Organizer
- U - User
- Null - No privileges

B.1.2 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>
```

B.1.3 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>
```

B.1.4 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>

```

B.1.5 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=<Virtual path of the application you want to create 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>

```

B.1.6 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=<Virtual path 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>

```

B.1.7 Revoking Access

Using the [REVOKEACCESS] script, you can remove users that are listed in the specified ACL.

```

[REVOKEACCESS]
APPLICATION=<Virtual path 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>

```

B.1.8 Creating Registries

Using the [REGISTRY] script, you can create registries.

```

[REGISTRY]
APPLICATION=<Virtual path of the application>
NAME=<Registry Variable Name>
VALUE=<Value for this variable>

```

B.1.9 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>
```

B.1.10 Deleting a User

Using the [DROPUSER] script, you can delete a user.

```
[DROPUSER]
NAME=<User Name>
```

B.1.11 Deleting a Group

Using the [DROPGROUP] script, you can delete a group.

```
[DROPGROUP]
NAME=<Group Name>
```

B.1.12 Deleting Access Privileges

Using the [DROPACL] script, you can delete access privileges provided to users.

```
[DROPACL]
APPLICATION=<Virtual path 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>
```

B.1.13 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>
```

B.1.14 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 With the WSH Tool

The WSH tool is a command-line tool that you can use to execute batch commands on the mobile server. Use the `-c` option to execute the commands you developed in the script INI file on the mobile server. The syntax for this option is as follows:

```
wsh -c <path_and_filename.ini> <username>/<password>[@<jdbc_url>]
```

Where:

- `<path_and_filename.ini>`: The INI file contains the batch commands for the mobile server. Provide the name and absolute path for the desired INI file.
- `<username>/<password>`: The mobile server repository administrator user name and password.
- `<jdbc_url>`: If the optional JDBC URL for the Oracle database that contains the mobile server repository is specified in the command line, then WSH will use this URL, else it defaults to use the URL configured in the `mobile.ora` file. You can specify the JDBC URL of a single Oracle database or an Oracle RAC database.
 - The JDBC URL for a single Oracle database has the structure of `jdbc:oracle:thin:@<host>:<port>:<SID>`.
 - The JDBC URL for an Oracle RAC database can have more than one address in it for multiple Oracle databases in the cluster and follows this URL structure:

```

jdbc:oracle:thin:@(DESCRIPTION=
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=TCP) (HOST=PRIMARY_NODE_HOSTNAME) (PORT=1521))
    (ADDRESS=(PROTOCOL=TCP) (HOST=SECONDARY_NODE_HOSTNAME) (PORT=1521))
  )
  (CONNECT_DATA=(SERVICE_NAME=DATABASE_SERVICENAME)))

```

For example, if you have an INI file called `mybatch.ini`, you can execute the WSH tool on this file. For this example, the mobile server repository user name and password is `mobileadmin/manager` and the JDBC URL (`<host>:<port>:<SID>`) is `myhost:1521:mySID`:

```
wsh -c mybatch.ini mobileadmin/manager@jdbc:oracle:thin:@myhost:1521:mySID
```

B.3 Examples of Batch Script Files for WSH

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=U
#
# Create group 'Sample Users' containing 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 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:
#[DROPUSE], [DROPGROUP], [DROPACL], [DROPREGISTRY],[DROPSNAPSHOTVAR].
#
# Dropuser JOHN
#
[DROPUSE]
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
```

Catalog Views for the Mobile Server

The following sections are a reference for the system catalog views for the Mobile Admin schema 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 C.1, "CV\\$ALL_CLIENTS"](#)
- [Section C.2, "CV\\$ALL_ERRORS"](#)
- [Section C.3, "CV\\$ALL_PUBLICATIONS"](#)
- [Section C.4, "CV\\$ALL_SUBSCRIPTIONS"](#)
- [Section C.5, "CV\\$ALL_SEQUENCES"](#)
- [Section C.6, "CV\\$ALL_SEQUENCE_PARTITIONS"](#)
- [Section C.7, "CV\\$ALL_PUBLICATION_ITEMS_ADDED"](#)
- [Section C.8, "CV\\$ALL_PUBLICATION_ITEMS"](#)
- [Section C.9, "CV\\$ALL_PUBLICATION_INDEXES"](#)
- [Section C.10, "CV\\$ALL_SUBSCRIPTION_PARAMS"](#)

C.1 CV\$ALL_CLIENTS

The CV\$ALL_CLIENTS view provides information about mobile clients.

[Table C-1](#) provides a description of ALL_CLIENT parameters.

Table C-1 ALL_CLIENTS Parameters

Column	Datatype	Null	Description
CLIENT	VARCHAR2 (60)	NOT NULL	The mobile client
LASTREFRESH_STARTTIME	VARCHAR2 (19)	NULL	Start time of the last refresh session
LASTREFRESH_ENDTIME	VARCHAR2 (19)	NULL	End time of the last refresh session

C.2 CV\$ALL_ERRORS

The CV\$ALL_ERRORS view provides information about failed client transactions.

[Table C-2](#) provides a description of ALL_ERRORS parameters.

Table C-2 ALL_ERRORS Parameters

Column	Datatype	Null	Description
CLIENT	VARCHAR2 (60)	NOT NULL	Client to which the failed transaction belongs.
TRANSACTION_ID	NUMBER (10)	NOT NULL	ID of the failed transaction.
ITEM_NAME	VARCHAR2 (30)	NULL	Name of the publication item that failed.
MESSAGE_TEXT	VARCHAR2 (2048)	NULL	Error text associated with the failed transaction and publication item.

C.3 CV\$ALL_PUBLICATIONS

The ALL_PUBLICATIONS view provides information about mobile server publications.

Table C-3 provides a description of ALL_PUBLICATIONS parameters.

Table C-3 ALL_PUBLICATIONS Parameters

Column	Datatype	Null	Description
NAME	VARCHAR2 (30)	NOT NULL	Publication Name.
TYPE	VARCHAR2 (40)	NULL	Publication Type.
NAME_TEMPLATE	VARCHAR2 (30)	NULL	Snapshot Name Template.
ENFORCE_RI	CHAR (1)	NOT NULL	Reserved.

C.4 CV\$ALL_SUBSCRIPTIONS

The ALL_SUBSCRIPTIONS view provides information about mobile server subscriptions.

Table C-4 provides a description of ALL_SUBSCRIPTIONS parameters.

Table C-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.

C.5 CV\$ALL_SEQUENCES

The ALL_SEQUENCES view provides information about mobile server sequences.

Table C-5 provides a description of ALL_SEQUENCES parameters.

Table C-5 ALL_SEQUENCES Parameters

Column	Datatype	Null	Description
NAME	VARCHAR2 (30)	NOT NULL	The sequence name.

C.6 CV\$ALL_SEQUENCE_PARTITIONS

The ALL_SEQUENCE_PARTITIONS view provides information about mobile server sequence partitions.

Table C-6 provides a description of ALL_SEQUENCE_PARTITIONS parameters.

Table C-6 ALL_SEQUENCE_PARTITIONS Parameters

Column	Datatype	Null	Description
CLIENT	VARCHAR2 (60)	NOT NULL	The client to which the sequence is assigned.
NAME	VARCHAR2 (30)	NOT 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.

C.7 CV\$ALL_PUBLICATION_ITEMS_ADDED

The ALL_PUBLICATION_ITEMS_ADDED view provides information about mobile server publication items.

Table C-7 provides a description of ALL_PUBLICATION_ITEMS_ADDED parameters.

Table C-7 ALL_PUBLICATION_ITEMS_ADDED Parameters

Column	Datatype	Null	Description
PUB_NAME	VARCHAR2 (30)	NOT NULL	The publication name.
ITEM_NAME	VARCHAR2 (30)	NOT 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	CLOB	NULL	The select statement.
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. You can only use fast refresh with a high priority restricting predicate. If you use any other type of refresh, the high priority restricting predicate is ignored.
WINNING_RULE	VARCHAR2 (30)	NULL	The winning rules option for resolving replication conflicts. Options include "client wins" and "server wins".

C.8 CV\$ALL_PUBLICATION_ITEMS

The ALL_PUBLICATION_ITEMS view provides information about mobile server publication items.

Table C-8 provides a description of ALL_PUBLICATION_ITEMS parameters.

Table C-8 ALL_PUBLICATION_ITEMS Parameters

Column	Datatype	Null	Description
NAME	VARCHAR2 (30)	NOT 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	CLOB	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. You can only use fast refresh with a high priority restricting predicate. If you use any other type of refresh, the high priority restricting predicate is ignored.

C.9 CV\$ALL_PUBLICATION_INDEXES

The ALL_PUBLICATION_INDEXES view provides information about mobile server publication item indexes.

Table C-9 provides a description of ALL_PUBLICATION_INDEXES parameters.

Table C-9 ALL_PUBLICATION_INDEXES Parameters

Column	Datatype	Null	Description
NAME	VARCHAR2 (30)	NOT NULL	Index name.
PUB_ITEM	VARCHAR2 (30)	NULL	Publication item name.
INDX_TYPE	CHAR (1)	NULL	Index type.
COLUMN_LIST	VARCHAR2 (2048)	NULL	Column list.

C.10 CV\$ALL_SUBSCRIPTION_PARAMS

The ALL_SUBSCRIPTION_PARAMS view provides information about mobile server subscription parameters.

Table C-10 provides a description of ALL_SUBSCRIPTION_PARAMS parameters.

Table C-10 *ALL_SUBSCRIPTION_PARAMS Parameters*

Column	Datatype	Null	Description
NAME	VARCHAR2 (30)	NOT NULL	Publication name.
CLIENT	VARCHAR2 (60)	NOT NULL	Client name.
PARAM_NAME	VARCHAR2 (30)	NOT NULL	Parameter name.
PARAM_VALUE	VARCHAR2 (62)	NULL	Parameter value.

iOS Mobile Device Management (MDM) Architecture

The following sections describe the Mobile Device Management (MDM) architecture and the device enrollment process:

- [Section D.1, "Description - MDM Architecture"](#)
- [Section D.2, "Device Enrollment Process"](#)
- [Section D.3, "Application Server Configuration for MDM"](#)

D.1 Description - MDM Architecture

See the following sections:

- [Section D.1.1, "MDM Command Cycle"](#)
- [Section D.1.2, "Additional Resources"](#)

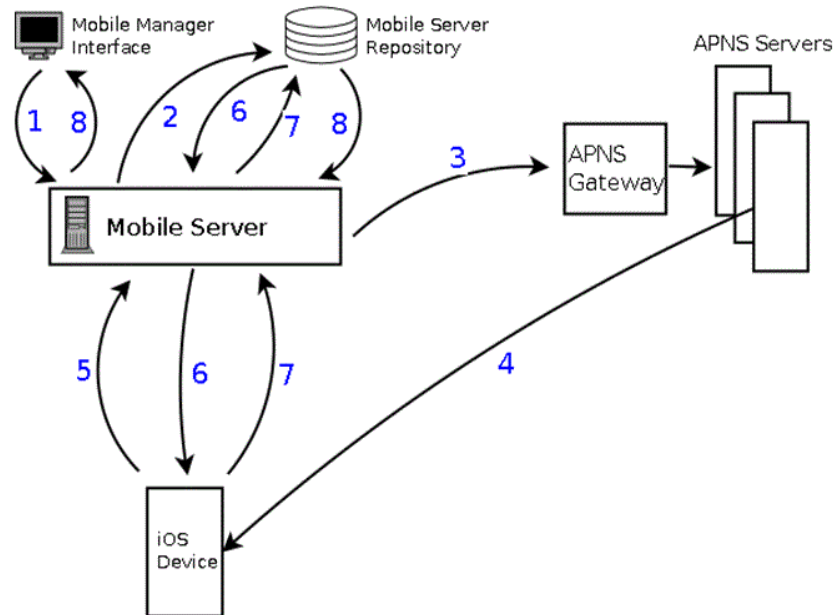
D.1.1 MDM Command Cycle

The main purpose of MDM protocol is to send commands to iOS devices and process the results. MDM protocol is based on HTTPS (secure HTTP) which is used for exchanging XML messages called property lists (plists). For more information on plists, see here:

<http://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/PropertyLists/Introduction/Introduction.html>

In MDM protocol, the server does not push commands to the device. Instead, the server uses Apple Push Notification Service (APNS) to notify the device that the new commands are available.

Once device receives push notification, it queries MDM server for commands, executes them and sends the results back to MDM server. [Figure D-1](#) shows how this works (for example, the command is **GetDeviceInfo**).

Figure D-1 Mobile Server MDM Architecture

The following steps explain how the MDM command cycle works:

1. User logs in to mobile manager via web interface, finds their device and issues command to it. Refer to [Section 7.11, "Using Mobile Manager to Manage iOS Devices"](#) for information on how iOS device management is integrated with Mobile Manager. See [Section 7.11.3, "iOS MDM Command Management"](#) for information on command management.
2. Mobile Server queues the command in the repository in command queue table.

Note: The other commands for the same device may be already queued as well.

3. Mobile Server contacts APNS Server via APNS Gateway and asks to send push notification to the device. APNS servers uniquely identify each device by a binary string called Device Token.
4. The push notification wakes up the device.
5. Device contacts Mobile Server to ask if a new command is available. In addition, the request can contain status and results of previous command already executed.
6. Mobile Server retrieves next queued command from command queue and sends it to the device.
7. The device executes the command and sends back to the server the command status - success or error (and error description) and the command result (which in our example would be device information).

Mobile Server will store both command status and device information received in its repository. If there are more queued commands available for this device the next queued command is retrieved from the repository and sent to the device as in step 6.

8. In Mobile Manager interface, the user can see the command history to verify that the command was executed successfully as well as device information which was retrieved by the command. For more information on Mobile Manager interface, see [Section 7.11.2, "View iOS Device Information"](#) and [Section 7.11.3, "iOS MDM Command Management"](#).

The cycle continues in this manner. The device executes commands, sends results to the server, asks for more commands and so on as long as more commands are available. If no more commands are available for this device, the device waits until another push notification is sent to it.

D.1.2 Additional Resources

On the internet, one of the most extensive documents provided on MDM is "MDM Protocol Reference". It describes in depth the interaction between MDM Server (in our case, Mobile Server), iOS devices and APNS. This information is helpful in case of any issues.

You need to have Enterprise account (iOS Developer Enterprise Program) to access MDM Protocol Reference document. Follow the steps below to get the document:

- Go to the Apple Developer's Portal, Member Center, and login with your Enterprise account credentials.
- Go to "Developer Centers" and click on "Mac" to go to Mac Developer's Center.
- Click on "Downloads". At the bottom right, you will see a link titled "View All Downloads". Click on this link. You will be taken to "Downloads for Apple Developers" page.
- Type "MDM" or "mobile" in the search box, and choose "Enter".
- Download "Mobile Device Management Protocol Reference" (you see that to the right in the search results).

D.2 Device Enrollment Process

Devices enroll in MDM when the server delivers a special configuration profile to the device. This profile contains:

- MDM Payload

This is a special payload that tells the device that it will be managed by the MDM server. It contains server URL, push notification topic and other attributes. For more details on MDM payload, see MDM Protocol Reference.

- Device Identity Certificate

Mobile Server needs to authenticate connected devices. Since MDM is done automatically without user interaction, a usual user name/password authentication will not work. Mobile Server (as MDM Server) authenticates devices by their identity certificates. This is called Client Certificate Authentication and is done in the server's SSL layer.

During enrollment, Mobile Server issues identity certificate for the device. Identity certificate is a X509 certificate/private key pair. It is included in the configuration profile in PKCS12 format. Each device certificate is signed by MDM CA (see [Section 11.2.1, "Creating and Configuring Certificate Authority"](#)). In addition, each certificate's Common Name (CN) contains unique device id (UDID).

After enrollment, when device connects to Mobile Server as part of MDM, server's SSL layer will verify that the device certificate is trusted. In addition, after SSL connection is established, Mobile Server will verify that the certificate was signed by MDM CA and that the UDID stored in the certificate's CN matches UDID in the message (each MDM message from the device contains UDID). For more information, see MDM Protocol Reference.

The enrollment process follows OTA (Over The Air) Profile Delivery and Configuration outlined by Apple. See here:

<https://developer.apple.com/library/ios/documentation/networkinginternet/conceptual/iphoneotaconfiguration/Introduction/Introduction.html>

- As described in [Section 7.11.1, "iOS Device Enrollment"](#), user provisions the device with MDM CA certificate. This is needed so that the device can accept certificates signed by this CA that will be provisioned in the following steps.

In addition, as described in [Section 11.2.2, "SSL Configuration for MDM,"](#) if the Mobile Server's SSL certificate was not obtained by trusted certificate authority, it should be signed by the MDM CA. In this case, having MDM CA certificate on the device will make the device trust the server's SSL certificate.

- User enters their credentials and clicks on the "Enroll" button (see [Section 7.11.1, "iOS Device Enrollment"](#)).
- Mobile Server receives user's credentials and authenticates the user. If authentication is successful, a special configuration profile is sent to the device asking for the device's UDID. This profile also contains a special challenge token.
- Device responds with its UDID and the challenge token it has received. The challenge token allows Mobile Server to associate UDID received with the user authenticated in the previous step. It also verifies that the server is talking to the device that was previously authenticated.
- Mobile Server generates another identity certificate and sends a configuration profile containing this identity certificate to the device. This is not the same profile or identity certificate used by MDM described above. This special certificate/private key pair is required by OTA process. This OTA certificate is also signed by MDM CA.
- Device sends back a response signed by the OTA private key received above. The response contains device's UDID.
- Mobile Server receives the response above and verifies it's signature. If the verification is successful, MDM configuration profile containing MDM Payload and Device Identity Certificate will be sent to the device.
- At this point the device is considered enrolled. A special command is issued that will install all uploaded Configuration and Provisioning profiles to the device and will also install all published applications assigned to the device's user. In addition, device information will be collected and stored in the repository.

This process mostly follows the diagram of OTA enrollment. See here:

https://developer.apple.com/library/ios/documentation/networkinginternet/conceptual/iphoneotaconfiguration/OTASecurity/OTASecurity.html#//apple_ref/doc/uid/TP40009505-CH3-SW1

One significant difference is that Mobile Server does not use SCEP protocol to sign identity certificates. Instead, the certificate/private key pair is generated by the Mobile Server and sent to the device as part of configuration profile in PKCS12 format. This is safe because all communication takes place over secure (SSL/HTTPS) connection.

D.3 Application Server Configuration for MDM

If you choose to enable MDM on your Mobile Server, additional configuration steps on your application server are performed. You do not need to perform these steps manually, they will be done automatically by the Mobile Server Repository Wizard (refer to step 15 of section 4.3.1.1, "Installation of the Mobile Server" of the *Mobile Server Installation Guide*). However, it may be useful to know the details in case you need to customize your Application Server.

By default, Mobile Server, when SSL is enabled, uses 1-way SSL protocol, where only the server presents its certificate to the client, but the client is not required to have or present certificate to the server (if client authentication is needed, it is done on an application level). However, MDM protocol uses 2-way SSL where both server and client present their certificates to each other, since the default way for the MDM client to be authenticated is via the client certificate.

Mobile Server supports both 2-way SSL (for iOS MDM) and 1-way SSL (for the rest of Mobile Server functionality) by using 2 different HTTPS listeners - one configured with 1-way SSL and the other with 2-way SSL. If you choose to enable MDM during Mobile Server installation, 2 SSL listeners will be configured each running on its own port. The default 1-way SSL listener is used for all Mobile Server functionality except iOS MDM (for example, sync, device management for non-Apple platforms, Mobile Manager access, and others). The 2-way SSL listener is used for iOS MDM exclusively.

The following sections show example configurations for supported application servers:

- [Section D.3.1, "Glassfish"](#)
- [Section D.3.2, "WebLogic"](#)

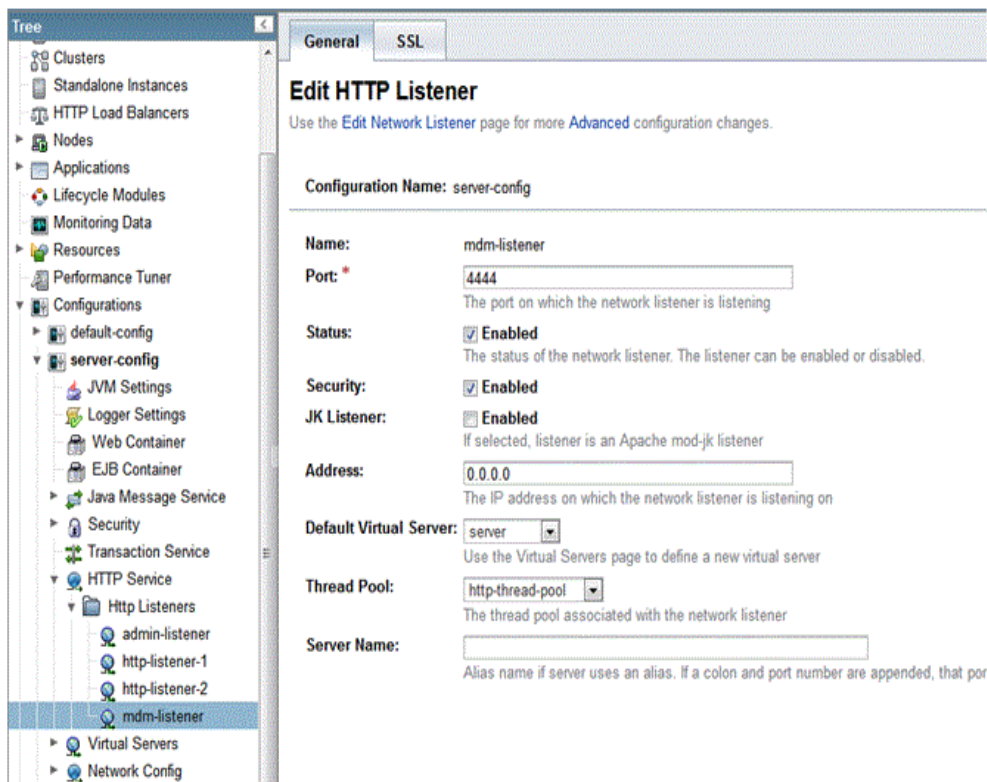
D.3.1 Glassfish

In Glassfish Administration Console, if you navigate under *Configuration->Server Config->HTTP Service->Http Listeners*, you see 3 HTTP listeners:

- HTTP listener for regular HTTP access
- HTTPS listener for SSL access to Mobile Server via 1-way SSL
- HTTPS listener with 2-way SSL used for MDM protocol

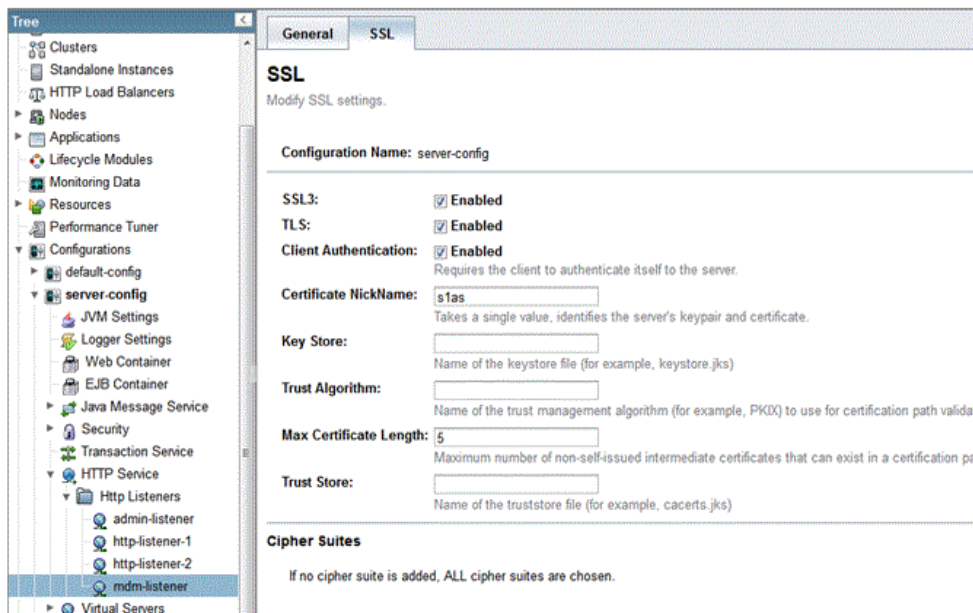
In [Figure D-2](#), they are named http-listener-1, http-listener-2 and mdm-listener respectively. All these listeners are running within the same server instance.

Figure D-2 Glassfish HTTP Listener Configuration



The "mdm-listener" is HTTPS listener with 2-way SSL enabled. You can see "Client Authentication" check-box selected in Figure D-3:

Figure D-3 Mdm-listener SSL Configuration



Note: The "Key Store" and "Trust Store" are not specified as this listener uses default Glassfish keystore and truststore located under GLASSFISH_HOME\domains\DMS_Domain\config directory in files keystore.jks and cacerts.jks respectively. These are the same keystore and truststore used by the 1-way SSL listener (http-listener-2 in our example). The "mdm-listener" also uses the same server identity (server private key/certificate pair) as the 1-way SSL listener.

This is specified by the same certificate alias (NickName).

If you decide to change the location of your keystore, truststore or server identity, be sure to specify the same for both SSL listeners.

D.3.2 WebLogic

In WebLogic, you can specify the main server ports called "Listen Port" and "SSL Listen Port" for HTTP and HTTPS respectively, as well as create additional HTTP or HTTPS listeners called "Channels" which are running within the same server instance.

In WebLogic Administration Console, if you navigate to your server's main configuration page (*Environment->Servers*, then click on your server), you will see SSL Listen Port specified, as in [Figure D-4](#):

Figure D-4 WebLogic Server Configuration Page

The screenshot shows the 'Settings for AdminServer' page in the WebLogic Administration Console. The 'General' tab is selected, and the 'Listen Port' and 'SSL Listen Port' fields are highlighted. The 'Listen Port' is set to 4777 and the 'SSL Listen Port' is set to 7002. Both are checked as 'Enabled'.

Field	Value	Description
Name	AdminServer	An alphanumeric name for this server.
Template	(No value specified) Change	Get the base server. More Info...
Machine	(None)	The WebLogic Server host on which this server runs. More Info...
Cluster	(Stand-Alone)	The cluster, or group of WebLogic Servers to which this server belongs. More Info...
Listen Address	<input type="text"/>	The IP address or DNS name of the server. More Info...
<input checked="" type="checkbox"/> Listen Port Enabled		Specifies whether this server (non-SSL) listen port. More Info...
Listen Port	<input type="text" value="4777"/>	The default TCP port that this server listens on for incoming connections. More Info...
<input checked="" type="checkbox"/> SSL Listen Port Enabled		Indicates whether the server listens on an SSL port. More Info...
SSL Listen Port	<input type="text" value="7002"/>	The TCP/IP port at which this server listens for SSL requests. More Info...

This is the default 1-way SSL Listen Port used for all of Mobile Server functionality except iOS MDM. To see the 2-way SSL Listen Port used for iOS MDM, click on "Protocols" and then choose "Channels". You see a channel called "mdm-channel" configured for HTTPS protocol, as shown in [Figure D-5](#):

Figure D-5 WebLogic Server Channels Page

Settings for AdminServer

Configuration **Protocols** Logging Debug Monitoring Control Deployments Services Security Notes

General HTTP jCOM IIOP **Channels**

Network channels allow you to manage quality of service, meet varying connection requirements, and improve utilization of your system.

This Network Channels page displays key information about each network channel that has been configured for this server.

[Customize this table](#)

Network Channels

New Clone Delete

<input type="checkbox"/>	Name	Protocol	Enabled	Listen Address	Listen Port
<input type="checkbox"/>	mdm-channel	https	true		7004

New Clone Delete

This channel runs on different port from the main server's SSL Listen Port. If you click on the "mdm-channel" and then click on "Security" tab, you will see that it is configured for 2-way SSL, as in [Figure D-6](#):

Figure D–6 Channel Security Configuration Page

Settings for mdm-channel

Configuration Monitoring

General Security

Save

This page allows you to define the security configuration of this network channel.

Two Way SSL Enabled Specifies whether th

Client Certificate Enforced Specifies whether cl
certificate authority

Channel Identity: Specifies the SSL ide

Server Private Key Alias: The string alias usec
keystore. This priva
certificate. [More I](#)

Custom Channel Private Key Alias:

Custom Channel Private Key Pass Phrase: The passphrase use
This passphrase is a
of null indicates that
the server's SSL cor

Confirm Custom Channel Private Key Pass Phrase:

Note: The "Two Way SSL Enabled" and "Client Certificate Enforced" check-boxes are selected, which means that valid client certificate is required to access this port.

This channel uses the same keystore, truststore and server identity as on the main SSL listen port.

Glossary

Base Table

A source of data, either a table or a view, that underlies a view. When you access data in a view, you are really accessing data from its base tables.

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.

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 provides a unique primary key values to the mobile client. 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 Development Kit

The Mobile Development Kit enables application developers to develop and debug applications.

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 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 troubleshooting purposes. ODBC support enables you to view the client's data, which is stored on a local database.

Oracle Database

The Oracle database is the database component of the mobile server.

Oracle Database 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.

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 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.

Sequence Window

The sequence window 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 window, the mobile client recreates the sequence with a new, unique range of values.

Snapshots

Snapshots are copies of application data that are captured 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.

The snapshots are automatically created on the client machine. Each subsequent time that a user goes connects through synchronization, the snapshots are either refreshed with the most recent data, or recreated 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.

Synchronization

Synchronization is the process used to replicate data between the mobile client and Oracle. The mobile server replicates (downloads) the user applications and data to the mobile client from the repository, 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.

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.

A

- access
 - defining access, 4-7
 - grant, 3-4, 4-11
 - revoke, 3-4, 4-11
- administrator
 - definition, 4-1
 - functions, 4-1
 - password, 9-1
 - privilege, 4-1
- alerts
 - synchronization, 5-10
- application
 - access, 4-1, 4-7
 - deleting, 3-2
 - grant access, 3-4, 4-11
 - properties
 - modify, 3-2
 - register database, 5-22
 - revoke access, 3-4, 4-11
 - scheduling to execute, 6-1
 - security, 4-7, 9-3
 - setting parameter values, 3-3, 4-14
 - user, 4-2
- apply
 - behavior, 5-4, A-7, A-9
- authentication
 - certificate rejection, 9-14
 - client, 9-17
 - external, 9-6
 - SSL, 9-17
- authorization, 4-7
- automatic synchronization, 5-13
 - platform rules, 5-14
 - conditions, 5-16
 - events, 5-15
 - network settings, 5-17

B

- Berkeley DB
 - encryption, 9-3

C

- cached user, 5-25
- catalog views
 - system, C-1
- certificate
 - authority
 - examples, 9-5
 - rejection, 9-14
 - self-signed, 9-14
- checkstatus command, 8-8
- cleanup command, 8-8
- client
 - clock, 5-5
 - device
 - delete, 7-5
 - dmagent, 7-24
 - non-SSL, 9-6
 - proxy, 9-15
 - software update request, 7-23
 - synchronization
 - GUI, 5-4, 5-5
 - updates
 - automatic, 7-4
 - using reverse proxy, 9-9
- clock, 5-5
- command
 - client pull, 7-25
 - create group commands, 7-18
 - create new, 7-17
 - disabling, 7-18
 - enabling, 7-18
 - history, 7-9, 7-19
 - in process, 7-9
 - input parameters, 7-16
 - modify, 7-16
 - reset password, 7-14
 - retrieve device information, 7-13
 - scheduling, 7-12, 7-14
 - sending, 7-9, 7-12
 - multiple devices, 7-12
 - single device, 7-12
- Common Access Card, 9-16, 9-17
- configuration, 9-19
 - platforms, 9-18
 - using, 9-20

- compose
 - behavior, 5-4, A-7, A-9
- conditions
 - automatic synchronization
 - platform rules, 5-16
- configuration
 - mobile server, A-1
 - mobile.ora, A-1
- connections
 - maximum concurrent, A-9
- CONSOLIDATOR section, A-6
- Consp perf utility, 5-39

D

- data
 - subsetting, 3-3, 4-14
- Data Synchronization, see synchronization
- database
 - password, 9-1
 - register for application, 5-22
 - restrict privileges, 9-1
 - users, 4-1
 - view information, 7-8
- DEBUG section, A-4
- device
 - add, 7-5
 - applying patches, 7-22
 - clock, 5-5
 - configure, 7-6
 - create commands, 1-7
 - customizing platforms, 1-7
 - delete, 7-5
 - disabling, 7-8, 7-11
 - enabling, 7-8, 7-11
 - management
 - client, 7-24
 - configuration, 7-3
 - modify configuration, 7-14
 - multiple users, 4-8
 - policy
 - user, 4-7
 - proxy server, 7-26
 - pull commands, 7-25
 - registration, 4-7
 - reset password, 7-14
 - retrieve
 - information, 7-13
 - retrieve file, 7-14
 - retrieve software information, 7-13
 - retrieve sync log, 7-13
 - sending commands, 7-9, 7-12
 - software
 - automatic update, 7-21
 - update, 7-19, 7-20, 7-26
 - software update, 7-22
 - stop, 7-13
 - swap user, 4-8
 - synchronize, 7-13
 - update, 7-14

- update software, 4-7
- view properties, 7-5
- viewing information, 1-7
- devmgr.inf file, 7-3
- devmgr.ini
 - configuring reverse proxy, 9-11
- DISABLE_PROMPT parameter, 7-4
- DISABLE_SSL_CHECK parameter, 9-14
- DISABLED_DML, 5-26
- dmagent, 5-8, 7-24
- DML
 - locks, A-13
- DO_APPLY_BFR_COMPOSE parameter, 5-4, A-7
- download
 - JAR or ZIP files, 7-33

E

- encryption, 9-3
 - password, 9-1
- Error queue
 - fixing synchronization errors, 5-34
 - synchronization, 5-3
- events
 - automatic synchronization, 5-15
- external authentication, 9-6

F

- farm
 - manage, 1-8
 - mobile server, 1-3, 4-4
 - synchronization configuration, 5-20
- file-based synchronization, 5-26
- FILESYSTEM section, A-3
- firewall
 - communication through proxy, 9-9
- force refresh
 - synchronization, 5-7

G

- group
 - access applications, 4-1
 - add, 4-11
 - add users, 4-12
 - delete, 4-11
 - grant application access, 4-13
 - managing, 4-1
 - remove users, 4-12
 - revoke access, 3-4, 4-11
 - revoke application access, 4-13
 - search, 4-5

H

- heap memory size, 5-12
- history
 - purging, 6-11
- HTTP
 - network protocol, 7-27

remote access, 9-1

I

IAS_MODE parameter, 4-9

id element

INF file, 7-23

In Queue

overview, 5-2

synchronization, 5-3

viewing transactions, 5-32

INF file

configuration, 7-10

description, 7-28

directory element, 7-32

env element, 7-33

execute section, 7-35

file element, 7-32

id element, 7-23

include element, 7-31

INI section, 7-34

install element, 7-31

keywords, 7-28

link element, 7-34

patch element, 7-22

register section, 7-35

registry element, 7-33

setup element, 7-29

user privileges, 7-29

version element, 7-23

install

distributing multiple clients, 8-1

distribution for multiple users, 8-8

Installation CD, 8-8

Installation Configuration File, see INF file

instance parameters

configuration, 5-20

internet

communication from inside intranet, 9-15

IP address, 5-12

J

JAR

download, 7-33

inflate, 7-33

Java

classpath, 5-12

job

creating, 6-5

using APIs, 6-11

define execution schedule, 6-6

definition, 6-1

delete, 6-10

disable, 6-3

disabling, 6-10

displaying history, 6-3

enabling, 6-10

history

purging, 6-11

managing, 6-3, 6-4

modifying, 6-7

scheduling, 6-1

Job engine

alerts, 6-3

Standalone, 6-2

Job Scheduler, 6-1

alerts, 6-3

engine management, 6-2

history, 6-3

managing active jobs, 6-3

start, 6-2

stop, 6-3

JVM

version on mobile server host, 5-12

K

keystore

creating, 9-4

keytool utility, 9-4

keytool utility, 9-4

keywords

INF file, 7-28

M

makedb command, 8-7

MAX_U_COUNT parameter, A-8

Message Generator and Processor, see MGP

MGP

alerts, 5-10

composing transaction, 5-3

create new MGP, 6-5

default process, 6-10

define execution schedule, 6-6

disabling, 6-10

enabling, 6-10

execution process, 5-3

managing, 6-4

modifying schedule, 6-7

overview, 5-2

purge history, 6-11

scheduled job, 6-1

statistics, 5-38, 6-8, 6-9

middle-tier

overview, 5-1

mobile client

configuring reverse proxy, 9-11

distributing multiple clients, 8-1

installation CD, 8-8

overview, 5-1

platforms, 7-9

proxy, 9-15, 9-16

viewing, 7-8

mobile device, see device

Mobile Manager

access, 4-1

active users, 10-2

how to use, 1-1

- log on, 1-1
- Upgradable parameter, 7-19
- MOBILE section, A-1
- mobile server
 - active sessions, 1-5
 - active users, 10-2
 - catalog views, C-1
 - configuring, A-1
 - reverse proxy, 9-11
 - SSL, 9-5
 - connect, 1-1
 - farm, 1-3, 1-8, 4-4
 - information, 1-2
 - list all, 1-2
 - manage, 1-1
 - applications, 1-5
 - users, 1-5
 - management tool, 1-1
 - overview, 5-1
 - password, 9-1
 - scripting language, B-1
 - example, B-5
 - SSL, 9-4
 - starting, 1-2
 - synchronization configuration, 5-20
 - tracing, 1-5
 - WSH tool, B-5
- mobile.ora, A-1
 - configuration, 5-20
 - configuring reverse proxy, 9-11
 - IAS_MODE parameter, 4-9
 - SSO_ENABLED parameter, 4-10
- msync
 - proxy, 9-16
- msync executable, 5-4, 5-5, 5-8
- multibyte characters
 - user name, 4-5, 4-6

N

- network
 - failure
 - automatic synchronization, 5-20
 - management, 7-27
 - protocol, 7-27
 - HTTP, 7-27
 - RAPI, 7-27
 - SMS, 7-27
 - provider
 - properties, 7-27
 - settings
 - automatic synchronization, 5-17
 - synchronizing when network unavailable, 5-26
- network provider, 5-7
- non-SSL
 - client, 9-6

O

ODBC

- driver
 - specify registration on client, 7-33
- odbc.ini
 - location, 7-2
- offline instantiation, 8-1
 - client directory, 8-2
 - configuration, 8-3
 - deploying, 8-8
 - MDK, 8-2
 - mobile server, 8-2
 - OLI engine, 8-6
 - overview, 8-1
 - tool, 8-3
- OID
 - migrating users, 4-9
 - migrating users from repository, 4-9
 - users, 4-4, 4-9
- oiduser tool, 4-9
- OLI
 - using makedb, 8-7
- oli.ini file, 8-3
- operating system
 - version on mobile server host, 5-12
- Oracle Internet Directory, see OID
- Oracle Single Sign
 - enable, 4-10
- Oracle Tag Language, see OTL
- ose.ini
 - location, 7-2
- OTL, 7-36
 - data types, 7-37
 - database connection, 7-44
 - DeviceManager, 7-47
 - HTTP request parameters, 7-45
 - HTTP session values, 7-45
 - operators, 7-40
 - sample, 7-49
- Out Queue
 - overview, 5-2
 - synchronization, 5-3
 - viewing subscriptions, 5-32

P

- package command, 8-7
- parameter
 - value, 3-3, 4-14
- password
 - allowed characters, 4-6
 - default account, 9-1
 - modify, 5-7
 - repository, 9-3
 - stored, 5-7
 - summary, 9-1
- patch
 - applying, 7-22
 - element, 7-22
- performance
 - analyzing synchronization, 5-39
 - Conspert utility, 5-39

- MGP, 5-38, 6-8, 6-9
- Sync Server, 5-37
- synchronization, 5-37
- platform
 - configure, 7-9
 - custom, 7-11
 - customize, 7-9
 - customized, 7-10
 - enable, 7-11
 - extend, 7-11
 - extending, 7-10
 - predefined, 7-9
 - rules
 - automatic synchronization, 5-14
- policy
 - setting attributes, 7-20
- privilege
 - administrator, 4-1
 - defining, 4-6
 - user, 4-2
- provisioning, 4-7
- proxy
 - reverse, 9-9, 9-15
 - server, 5-7
- proxy server
 - device, 7-26
- publication item
 - setting parameter values, 3-3, 4-14
 - view as listed in repository, 5-32
- publications
 - performance, 5-39
 - view as listed in repository, 5-31
- publishing, 4-1

Q

- queues
 - involved in synchronization, 5-3

R

- RAPI
 - network protocol, 7-27
- reports
 - system status, 10-1
- repository
 - browsing publication items, 5-29
 - browsing publications, 5-29
 - browsing synchronization queues, 5-29
 - browsing users, 5-29
 - migrating users to OID, 4-9
 - password, 9-1, 9-3
- Resource Manager
 - setting attributes, 7-20
 - UPDATE_SOFTWARE attribute, 7-20
- resume
 - synchronization, 5-20
- resume file, A-9
 - sizing, A-9
- RESUME_BLOCKSIZE parameter, 5-21

- RESUME_FILE parameter, 5-21, A-9
- RESUME_FILE_SIZE parameter, 5-21, A-9
- RESUME_MAX_WAIT parameter, 5-21, A-9
- RESUME_MAXACTIVE parameter, 5-21
 - synchronization
 - resuming interrupted, A-9
- RESUME_MAXCHUNK parameter, 5-22, A-9
- RESUME_TIMEOUT parameter, 5-21, A-9
- retry
 - automatic synchronization, 5-20
- reverse proxy, 9-9, 9-15
 - configuring mobile client, 9-11
 - SSL, 9-12
- REVERSE_PROXY parameter, A-3
- reverse_proxy parameter, 9-11

S

- scripting language, B-1
 - example, B-5
 - execution of INI file, B-5
 - mobile server, B-1
- security, 4-7
 - authentication, 9-1, 9-6
 - designing application, 9-3
 - firewall, 9-1, 9-9
 - manage, 9-1
 - mobile client, 9-1, 9-3
 - password, 9-1
 - repository password, 9-3
 - restrict database privileges, 9-1
 - reverse proxy, 9-1, 9-9
 - SSL, 9-1, 9-4
 - client authentication, 9-1, 9-16
- server
 - controlling load, 5-21
 - set max connections, 5-21
- SERVER_URL parameter
 - reverse proxy, 9-11
- sessions
 - active users, 10-2
- setAttribute method, 7-20
- shared parameters
 - configuration, 5-20
- SKIP_INQ_CHK_BFR_COMPOSE parameter, 5-4, A-9
- SLEEP_TIME parameter, A-10
- smartcard, 9-17
- SMS
 - network protocol, 7-27
- software
 - automatic update, 7-21
 - request update, 7-23
 - update, 5-6, 7-19, 7-20, 7-22, 7-26
 - viewing installed, 7-9
- SQLite
 - encryption, 9-3
- SSL, 9-4
 - client authentication, 9-16
 - Common Access Card, 9-16, 9-17

- configuring mobile server, 9-5
- creating keystore, 9-4
- mixing secure and non-secure sites, 9-6
- No available certificate, 9-6
- reverse proxy, 9-12
- smartcard, 9-17
- troubleshooting, 9-6
- SSO_ENABLED parameter, 4-10
- Standalone Job engine, 6-2
- subscriptions
 - profiling, 5-39
- Sync Agent
 - manages automatic synchronization, 5-19
- Sync Client
 - downloading data, 5-3
- Sync client
 - overview, 5-2
- Sync Server
 - alerts, 5-10
 - execution process, 5-3
 - history, 5-11
 - managing, 5-9
 - overview, 5-2
 - start, 5-9
 - statistics, 5-37
 - uploading data, 5-3
 - user sessions, 5-11
- SYNC_HISTORY parameter, 5-11
- synchronization
 - alerts, 5-10
 - analyzing performance, 5-39
 - automatic
 - retry after failure, 5-20
 - Sync Agent, 5-19
 - browsing repository, 5-29
 - client GUI, 5-4, 5-5
 - composing transaction, 5-3
 - configuration
 - instance parameters, 5-20
 - shared parameters, 5-20
 - conflict, 5-34
 - controlling server load, 5-21
 - correcting errors, 5-34
 - DML locks, A-13
 - downloading data, 5-3
 - error
 - re-execute transaction, 5-34
 - execution steps, 5-3
 - file-based, 5-26
 - create file, 5-27
 - troubleshooting, 5-28
 - force refresh, 5-7
 - history, 5-11
 - init.ora parameters, A-13
 - Linux, 5-8
 - monitor with SQL scripts, 5-40
 - no network, 5-26
 - options, 5-7
 - overview, 5-1
 - performance, 5-37

- priority, 5-7
- purge history, 6-11
- queues, 5-3
 - view, 5-32
- resuming interrupted, 5-20
- separate databases, 5-22
- sharing data among clients, 5-25
- temporary data storage, 5-21
- tracing, 5-29
- uploading data, 5-3
- system
 - status, 10-1

T

- time zone, 5-5
- timeout
 - client, A-9

U

- update
 - software for device, 4-7
- update utility, 5-8
- UPDATE_SOFTWARE attribute, 7-20
- UPDATE_SOFTWARE_APPS attribute, 7-20
 - Resource Manager
 - UPDATE_SOFTWARE_APPS attribute, 7-20
- UPDATE_SOFTWARE_PLATFORM attribute, 7-20
- updating software, 7-22
- Upgradable parameter, 7-19
- upgrade
 - software, 7-22
- user
 - access applications, 4-1
 - active, 10-2
 - add new, 4-5
 - administrator, 4-1
 - allowed characters in name, 4-6
 - application user, 4-2
 - authentication, 5-5, 5-7
 - authorization, 4-7
 - cached, 5-25
 - database users, 4-1
 - definition, 4-1
 - delete, 4-11
 - device policy, 4-7
 - display, 4-4
 - display name, 4-5
 - duplicating, 4-8
 - enabled, 4-4
 - install privileges, 7-29
 - listed in repository, 5-29
 - managing, 4-1
 - migrating to OID, 4-9
 - name, 4-5
 - OID, 4-4, 4-9
 - password, 4-5, 9-1
 - privilege, 4-6
 - privileges, 4-1, 4-6

- properties, 4-5
- repository, 4-4
- revoke access, 3-4, 4-11
- search, 4-5
- software update, 7-22
- swap, 4-8
- types of, 4-1
- user name
 - multibyte characters, 4-5, 4-6

V

- variable
 - setting, 3-3, 4-14
- version element, 7-23

W

- Workspace
 - Mobile Manager, 1-1
- WSH tool
 - description, B-5
 - overview, B-1
 - scripting language, B-1

Z

- ZIP
 - download, 7-33
 - inflate, 7-33

