

Oracle Utilities Network Management System

Operations Mobile Application

Installation and Deployment Guide

Release 2.5.0.1.5

F41246-05

February 2022

F41246-05

Copyright © 1991, 2022 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 embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are “commercial computer software” or “commercial computer software documentation” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government’s use of Oracle cloud services are defined by the applicable contract for such services. 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 Inside 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, Epyc, and the AMD 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	1-vii
Audience.....	1- vii
Related Documents	1- vii
Conventions	1- viii
 Chapter 1	
Installation and Deployment Overview	1-1
Server Installation Overview	1- 1
Client Development Installation Overview	1-2
Client Configuration Overview	1- 2
Client Deployment Overview	1- 2
 Chapter 2	
Supported Platforms & Hardware Requirements	2-1
Hardware Requirements	2- 1
Client Hardware Requirements	2-1
Server Hardware Requirements	2-2
Development Hardware Requirements	2-2
Prerequisite Software.....	2- 2
 Chapter 3	
Mobile Gateway Server Installation	3-1
Mobile Gateway Architecture	3- 1
Deploy the Mobile Gateway	3-3
Configuring WebLogic to Handle HTTP Basic Challenges Correctly	3-7
Configuring the Default Control Zone and Crew Defaults	3- 7
 Chapter 4	
NMS Server Configuration	4-1
GeoJSON Map Generation.....	4- 1
Overview	4-1
Directory Location	4-1
Build Processes	4-2
GeoJSON Configuration File	4-2
GeoJSON Map Deployment	4-3
GeoJSON Offline Landbase Maps	4-3
Mobile User Validation	4-4
Permissions and Permission Sets	4-5
Predefined Users	4-5
Creating Users Using a Key	4-5

Using LDAP/AD User Validation	4-6
Mobile Applications Validation	4-7
Configuring OMA Highlight Coloring	4-8
Server Based Documents	4-9
Overview	4-9
Document Locations	4-9
Document Configuration in OMA.....	4-9
Configuring OMA Object Attribute Viewer	4-10
Overview	4-10
Configuration.....	4-10
Configuring OMA For Schematic Maps	4-12
Overview	4-12
Configuration.....	4-12
Configuring OMA Search Options	4-13
Overview	4-13
Configuration.....	4-13

Chapter 5

Client Development Setup	5-1
Install Software	5-1
Install Prerequisite Software	5-1
Install Operations Mobile App SDK.....	5-1
Build Operations Mobile Application.....	5-2
Testing.....	5-2
Test with a Browser	5-2
Test with iOS Device	5-2
Test with Android Device	5-2
Test with Windows 10.....	5-2
OMA Client Configuration.....	5-3
Third Party Application Integration with the OMA Client.....	5-3

Chapter 6

Client Deployment	6-1
Android.....	6-1
Google Play Store	6-1
Alternative Distribution Options.....	6-1
Pre-Installed Devices.....	6-1
IT Installation Service.....	6-1
iOS.....	6-2
App Store	6-2
iOS Developer Enterprise Program.....	6-2
Custom B2B Apps Program.....	6-2
Ad Hoc Distribution (intended for Testing)	6-2
iOS Beta Testing Service: TestFlight.....	6-2
Windows.....	6-3
Windows Store.....	6-3
Alternative Distribution Options.....	6-3
Pre-Installed Devices.....	6-3
IT Installation Service.....	6-3

Chapter 7

Operations Mobile Application Setup on OPAL	7-1
--	------------

Chapter 8

Operations Mobile Application Project Setup 8-1

Appendix A

Restricted Use and User License Terms A-1
 Mobile Archive Restricted Use..... A- 1

Preface

The information in this document is intended to guide you through a successful implementation and deployment of the Oracle Utilities Network Management System Operations Mobile Application.

This preface contains these topics:

- [Audience](#)
- [Related Documents](#)
- [Conventions](#)

Audience

This document is intended for anyone responsible for implementing the Oracle Utilities Network Management System Operations Mobile Application.

Related Documents

For more information, see the following documents in the Oracle Utilities Network Management System Release 2.5.0.0.0 documentation set:

- *Oracle Utilities Network Management System Adapters Guide*
- *Oracle Utilities Network Management System Advanced Distribution Management System Implementation Guide*
- *Oracle Utilities Network Management System Configuration Guide*
- *Oracle Utilities Network Management System for Water User's Guide*
- *Oracle Utilities Network Management System Installation Guide*
- *Oracle Utilities Network Management System Licensing Information User Manual*
- *Oracle Utilities Network Management System Quick Install Guide*
- *Oracle Utilities Network Management System Release Notes*
- *Oracle Utilities Network Management System Security Guide*
- *Oracle Utilities Network Management System User's Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
<code>monospace</code>	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Chapter 1

Installation and Deployment Overview

- [Server Installation Overview](#)
- [Client Development Installation Overview](#)
- [Client Configuration Overview](#)
- [Client Deployment Overview](#)

The Oracle Network Management System Operations Mobile Application (or App) is delivered as two components:

1. The server side Mobile Gateway. This gateway must be installed on an application server available to the clients. If the clients are coming in from the public internet, this Mobile Gateway must be available on the public internet. This Mobile Gateway will then interface to the Oracle Network Management System application server based on firewall/network configurations setup by your IT staff.
2. The Oracle Network Management Systems Operations Mobile Application Software Development Kit (Operations Mobile Application/SDK), which contains the source code of the mobile application. The Operations Mobile Application/SDK must be compiled to the target platform and installed on the platforms in order to run.

If you are using the Mobile Gateway for an interface or service (other than OMA or another mobile client), where the service is acting on behalf of users, please refer to the *Oracle Utilities Network Management System Adapters Guide* REST API chapter, and note the `as-user` parameter on many of the APIs. This parameter will allow you to specify the user who performed the work rather than the service that reported the work via the REST API.

Server Installation Overview

Follow these steps to install, build and deploy the Oracle Network Management System Operations Mobile Application:

1. Install and configure the Oracle Network Management System as described in the *Oracle Utilities Network Management System Installation Guide*.
2. Install the Oracle Network Management Systems Mobile Gateway server as defined in the section Mobile Gateway Server Installation.
3. Configure the model requirement of the mobile app as defined in the section GeoJSON Map Generation.

Client Development Installation Overview

Follow these steps to install, build and deploy the Oracle Network Management System Operations Mobile Application:

1. Decide the client platforms you plan on supporting, use the table in the Hardware Requirements section to identify the build environment platform that supports your targeted clients. It is recommended to build the browser platform of the application for testing and system verification.
2. Review and prepare for the download and installation of required Oracle and third-party software as described in the section Prerequisite Software.
3. Install the third-party software.
4. Unzip the Oracle Network Management System Operations Mobile Application project files from the \$NMS_BASE/sdk/OMA2.zip file to your build environment system.
5. Unzip the Oracle Network Management System Operations Mobile Application 3rd party files from the \$NMS_BASE/sdk/OMA2.3rdparty.zip file to your build environment system.
6. Install the Node.JS from <https://nodejs.org>. You will need version 10.18 or higher.
7. Build the Oracle Network Management Systems Operations Mobile App for each of the desired platforms.
8. Run the compiled client using the browser to test your built application.
9. Run the compiled client using the target hardware platform in development mode.

Client Configuration Overview

The Configuration of the client consists of the following:

1. Installing your GeoJSON offline landbase maps and index
2. Setting you default server URI in src/js/resources/config/loginSettings.js
3. Creating symbol files (.svg)
4. Mapping map objects to symbols (devices, conductors, conditions).
5. Configure all resources and resources/config files.

Client Deployment Overview

The Deployment of the client will be based on a number of constraints:

1. The target client platform.
2. IT installation vs end user Installation

Please work with your IT department to identify the best deployment method.

Chapter 2

Supported Platforms & Hardware Requirements

- [Hardware Requirements](#)
- [Prerequisite Software](#)

Hardware Requirements

Client Hardware Requirements

The following are the hardware requirements for the mobile application client:

Client	Version
iOS Tablets or Phones	iOS devices running iOS 14.x or greater.
Android Tablets or Phones	Android device running Android 6.0.1 or greater.
Windows PC or Tablet	Windows device running Window 10 version 1903 or above and a high performance CPU such as Intel i3, i5, or i7; Arm or Atom CPUs are not recommended.
Web Browser (The web browser is available for testing purposes only; not all functionality and security are supported.) ^a	Chrome browser running 40.0 or newer running on Windows or Mac Hardware. Safari browser running on Mac OSX. Firefox running on Windows. Edge running on Windows.

a.Web Browser - Limited Support

The use of web browsers listed above is not support or production use. Browsers have ever changing security vulnerabilities and rules that make it difficult to support them reliably. Here are some of the OMA functions that may not be support by all the browsers, there may be more not listed:

Map downloads (works on Firefox).

Getting location latitude/longitude (if serving OMA from an http source).

Taking pictures.

Attaching files/picture.

- Off-line mode (simulated with online/offline button in upper right header bar).
- User Interface scaling

Server Hardware Requirements

The following are the hardware requirements for the mobile application server:

Application Server

An Oracle WebLogic application server is required to deploy the nms-ws.ear file that is included in the Oracle Network Management System release package. The WebLogic version must match the version used for the Oracle Network Management System cesejb.ear. The nms-ws application server requires a minimum of 2 CPU cores and 8 GB of memory.

Development Hardware Requirements

The following are the hardware requirements for the development of the mobile application client:

Build Environment Hardware	Android Devices	iOS Devices	Windows PC or Tablet	Web Browser
Windows	Yes	No	Yes (Windows 10 or higher)	Yes
Macintosh OS X	Yes	Yes	No	Yes
Linux	Yes	No	No	Yes

Prerequisite Software

The following software must be installed and configured prior to installation of the Oracle Network Management System Operations Mobile Application Software Development Kit:

- Node.js (v14.7.3+), a platform built on Chrome's JavaScript runtime for building fast, scalable network applications.
- npm 7.23.0 or higher.
- Oracle WebLogic 12g for the NMS Mobile Gateway
- Android Studio (v3.4.1+) for creating new applications for the Android operating system.
- XCode 12.4 or greater, running on MacOS 11.6 (Big Sur) or newer, and Apple Developer ID for iOS 14.x devices for developing the applications using iOS SDK.
- Microsoft Visual Studio 2017 or above.

The Oracle JET (JavaScript Extension Toolkit) website has many resources to help you learn the Oracle JET development processes: <https://www.oracle.com/webfolder/technetwork/jet/index.html>.

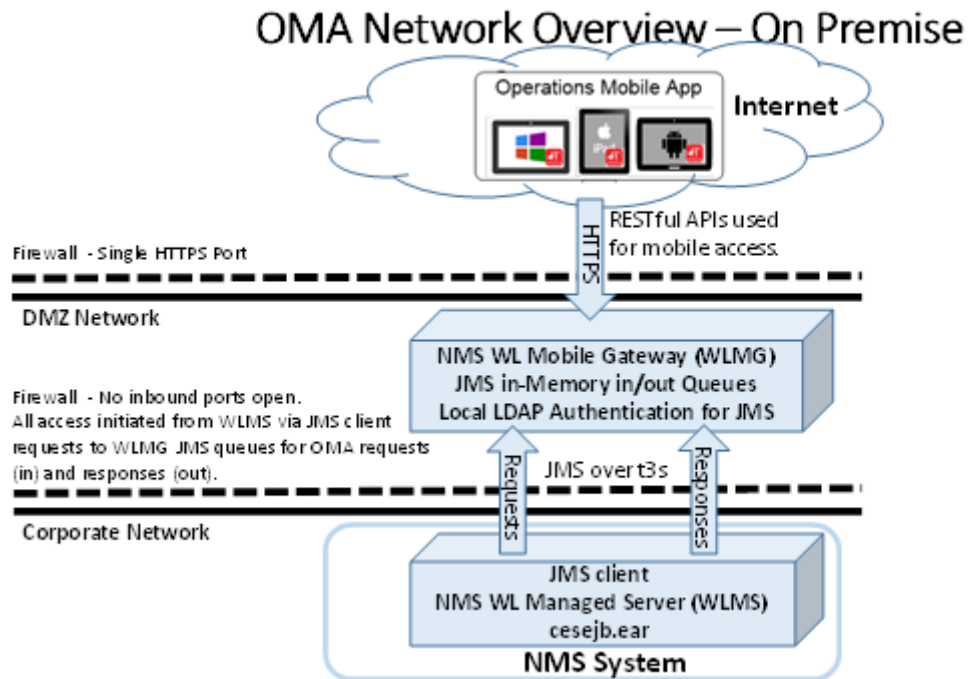
Chapter 3

Mobile Gateway Server Installation

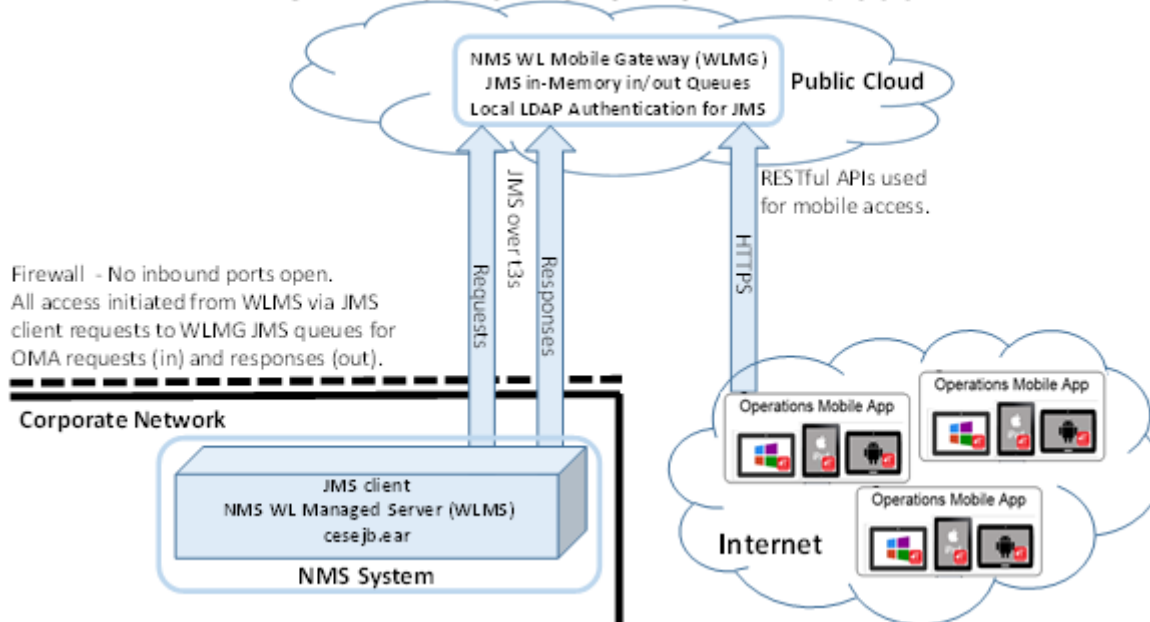
- Deploy the Mobile Gateway
- Configuring WebLogic to Handle HTTP Basic Challenges Correctly

Mobile Gateway Architecture

The Mobile Gateway can be deployed on premise or in the cloud:



OMA Network Overview – In Cloud



This architecture addresses many security concerns limiting the access from the Internet to the corporate network.

How Operations Mobile App devices connect to the NMS instance:

1. The Operations Mobile App client device connects to a dedicated WebLogic Managed Server, called the NMS WebLogic Mobile Gateway (WLMG), using HTTPS making RESTful Web Service requests.
2. The WLMG places Operations Mobile App client requests on the JMS in-memory "requests" queue.
3. The primary NMS WebLogic Managed Server (WLMS) connects to the JMS "requests" queue on the WLMG as a JMS client using the WebLogic t3s protocol and pulls requests off the "requests" queue and processes the requests via the normal channel from WLMS to NMS Services.
4. The WLMS then places responses to valid Operations Mobile App client requests on the parallel JMS in-memory "responses" queue – in a similar fashion to how the "requests" queue is handled.
5. The WLMG replays to the HTTPS RESTful Web Services request with the WLMS responses
6. The Operations Mobile App client device takes the HTTPS response and processes it on the device.

Notes:

- It is recommended to have an Oracle HTTP server or other reverse proxy server to further isolate OMA from the internet.
- There should be a firewall rule that allows only access to the https port from the internet (all other ports should be blocked).

Deploy the Mobile Gateway

The Oracle Network Management System Mobile Gateway is delivered in the \$NMS_BASE/dist/install directory and when the nms-install-config --java is run, the deployable Oracle Network Management System Mobile Gateway will reside in \$NMS_BASE/java/deploy/nms-ws.ear.

The Oracle Network Management System Mobile Gateway nms-ws.ear is deployed to the target WebLogic server. The nms-ws.ear expects a proxy user to connect between the nmw-ws.ear and the cesejb.ear, the default is `mobile-proxy` and it needs to be included in the Role `NmsMobile`. If you are configuring multiple authentication providers, please mark them as `Control Flag = OPTIONAL`.

The following changes should be done on the main NMS domain, as well as the mobile gateway domain (if you are using separate domains).

1. Uncomment the following line in \$NMS_CONFIG/jconfig/build.properties, modifying the user if necessary:

```
config.ws_runas_user = mobile-proxy
```

2. In the WebLogic console, do the following:
 - In the Domain Structure, click the Summary of Security Realms link.
 - On the Summary of Security Realms page, click **myrealm**.
 - On the Settings for myrealm page, click the **Users and Groups** tab.
 - In the Users tab, click **New** and create the username matching the proxy username you are using.
 - On the Settings for myrealm page, click the **Roles and Policies** tab.
 - In the Roles table, expand Global Roles
 - Click **Roles**.
 - On the Global Roles page, click **New**.
 - On the Create a New Role for this Realm page, enter `NmsMobile` in the Name field. The role will be listed in the Global Roles table.
 - Click the **NmsMobile** role name link to edit the role.
 - On the Edit Global Role page, under Role Conditions, add the user `mobile-proxy` (or whatever proxy user name you are using). The proxy user should **not** be a member of any group.)
3. On the server that is running nms-ws.ear, run:

```
wlst oma-jms.py
```

The script will prompt you for login credentials for the nms-ws WebLogic admin server, the server name or cluster name, and a suffix to add to each of the elements being created. The script will create the following:

- A JMS Server called JMSServer-oma
- A JMS system module called SystemModule-oma

The SystemModule-oma defines a connection factory called `MobileConnectionFactory`, with the following changes from the default:

- The JNDI name is `jms/MobileConnectionFactory`
- The Default delivery mode is Non-Persistent
- The default time to live is 300000

- The client acknowledge policy is *Previous*
- The Flow Control Enabled is checked
- The One Way Send Mode is “Queue or Topic”
- The security is set to only allow access from a user with the NmsMobile role

It will also create the oma-to-nms queue with the JNDI name jms/oma-to-nms, and the nms-to-oma queue with the name jms/nms-to-oma.

The factory and the queues are deployed to a subdeployment to the JMSServeroma.

4. Run `nms-install-config --java`, which will rebuild the ear files with the configured username and install the nms-ws.ear.

The server running nms-ws.ear needs to have a SSL certificate configured. Follow the steps in the *Oracle Utilities Network Management System Installation Guide* “Configure Keystores” section.

If the nms-ws.ear file is deployed on the same WebLogic managed server as the Oracle Network Management System cesejb.ear file, there is no additional configuration required; however, if the nms-ws.ear file is deployed on a different WebLogic managed server than the cesejb.ear in the same domain or a different domain, you will need to do the remaining steps in this section.

5. Do the following on the WebLogic domain where nms-ws will be deployed:

- Define a proxy user and role as you did for the main NMS server.

Note: For the best security, use a different password than you used for the proxy user on the main NMS server.

- Set the domain Trust Credential

- Select the domain name at the top of the Domain Structure panel.
- On the Settings page for the domain, click the Security tab and its General sub-tab.
- At the bottom of the panel, expand the Advanced settings and enter the Credential and Confirm Credential value you plan to establish domain trust.

- Settings for the managed server:

Note: If deploying in a WebLogic cluster, repeat these steps for each managed server in the cluster.

- Configuration tab/General sub-tab Listening Port Enabled not checked and SSL Listening Port Enabled checked with a valid SSL Listen Port specified.
- Protocols tab/General sub-tab Enable Tunneling checked.
- Protocols tab/Channels sub-tab. Create a new channel for JMS queue communications for the NMS WebLogic server. Click **New** and enter:
 - **Name:** enter a name (OMA-JMS-Channel)
 - **Protocol:** Select t3s or https.

Note: Some corporate firewalls will not allow t3s communication to the external system and will require https. However, if the Mobile Gateway will be deployed to a WebLogic cluster, using t3s is recommended over https for the JMS connection between the NMS WebLogic instance and the Mobile Gateway cluster. This is because t3s supports configuration of multiple WebLogic cluster members directly, and failover is handled within WebLogic. Using https requires the specification of a single host address through an external load balancer. A properly configured load

balancer should then (in turn) find an available WebLogic cluster member as long as one is available.

- Click **Next**.
 - **Listen Address:** use the same address as the main listen address for the managed server default listen address.
 - **Listen Port:** use an unused port on the managed server.
 - **External Listen Address:** use the public facing DNS known host name. If the host does not have a public facing DNS known hostname, use the public facing IP address.
 - **External Listen Port:** Use the public facing port for this channel
 - Click **Next**.
 - Check all four options: **Enabled, Tunneling Enabled, HTTP Enabled for This Protocol,** and **Outbound Enabled**.
 - Click **Finished**.
- If the managed server is on a cloud system with a network controlled front end, such as the Oracle Java Cloud Service, be sure to expose the default https port and the JMS queue channel port to the public internet.
6. In the WebLogic console, under Work Managers, add a new work manager called nms-ws-work-manager, and accept the defaults. Then edit the manager, and add a new "Maximum Thread Constraint", and choose as a default 80. (This may need to be adjusted but as a starting point use 5 times the number of cpu threads). If you see the managed server requiring too much memory, you can lower this value. If there is cpu and memory to spare, you can increase it up to 200 maximum.
7. Do the following on the NMS Managed Server:
- Verify the managed servers running ceselj.ear have the following in their system startup parameters: `-Djava.awt.headless=true`
 - Create a new System Module.
 - In that new system module, create a New Foreign server named nms-ws and accept the defaults.
 - Select the new system module and click the Security tab's Policies sub-tab.
 - Add Conditions, Role, and add NmsMobile.
 - Click **Finish**.
 - Select nms-ws and configure the following:
 - **JNDI Initial Context Factory:**
weblogic.jndi.WLInitialContextFactory
 - **JNDI Connection URL:** The URL to the gateway server, where the server name and port match the hostname(s) and port(s) defined above when configuring OMA-JMS-Channel for each OMA managed server. The URL should be in the format `t3s://somehost:port`.

Notes:

- If clustering, the URL should be specified as `t3s://server1:port1,server2:port2,server3:port3`
- Some corporate firewalls will not allow t3s communications to external system and will require https. If this is the case, use https instead of t3s. However, t3s is recommended for cluster failover.
- If the nms-ws managed server is not on a host with a known DNS name, use the public IP address for somehost.
- **JNDI Properties Credentials:** The password of the mobile-proxy user.
- **JNDI Properties:**
`java.naming.security.principal=mobile-proxy` (replace mobile-proxy with the name of the mobile-proxy user)
- **Default Targeting Enabled** should be checked.
- Click **Save**.
- Under the **Destinations** tab, create a new destination:
 - **Name:** oma-to-nms
 - **Local JNDI Name:** `jms/oma-to-nms`
 - **Remote JNDI Name:** `jms/oma-to-nms`
- Under the **Destinations** tab, create another new destination:
 - **Name:** nms-to-oma
 - **Local JNDI Name:** `jms/nms-to-oma`
 - **Remote JNDI Name:** `jms/nms-to-oma`
- Under **Connection Factories**, create a new factory:
 - **Name:** `MobileConnectionFactory`
 - **Local JNDI Name:** `jms/ MobileConnectionFactory`
 - **RemoteJNDI Name:** `jms/MobileConnectionFactory`
- Set the domain Trust Credential.
 - Select the domain name at the top of the **Domain Structure** panel.
 - On the **Settings** page for the domain, click the **Security** tab and its **General** sub-tab.
 - At the bottom of the panel, expand the **Advanced** settings and enter the **Credential** and **Confirm Credential** value you plan to establish domain trust.

Configuring WebLogic to Handle HTTP Basic Challenges Correctly

By default, WebLogic attempts to intercept all HTTP Basic Authentication challenges. This default behavior needs to be disabled for the WebLogic domain where the nms-ws.ear is deployed for the Oracle Network Management System Operations Mobile Application to function correctly.

See your WebLogic documentation for the location of `config.xml`, the WebLogic configuration file.

Add the `<enforce-valid-basic-auth-credentials>` element to `config.xml` within the `<security-configuration>` element. The edited file should look like the following:

```
...
<enforce-valid-basic-auth-credentials>false</enforce-valid-basic-auth-
credentials>
</security-configuration>
...
```

Save the updated `config.xml` file and restart WebLogic (if it is running).

Configuring the Default Control Zone and Crew Defaults

To configure the control zone and crew defaults, edit `jconfig/server/WebService.properties`.

Change the `default_mobile_control_zone` to the zone that you wish the automatically created crews to use.

If crew groups are used, define the following in `WebService.properties` (changing the value as appropriate):

```
default_mobile_crew_center = Mobile Crew Center
```

If crew centers are used, define the following in `WebService.properties` (changing the value as appropriate):

```
default_mobile_crew_group = Mobile Group
```

Chapter 4

NMS Server Configuration

- GeoJSON Map Generation
- Mobile User Validation
- Server Based Documents
- Configuring OMA Object Attribute Viewer
- Configuring OMA Search Options

GeoJSON Map Generation

Overview

The Oracle Network Management System Operations Mobile App uses electrical facility maps in GeoJSON format (see <http://geojson.org> for details). Oracle Network Management System provides tools and scripts (collectively referred to as the GeoJSON generator) to build GeoJSON versions of your electrical facility maps. In addition to electrical facility map, OMA can support GeoJSON Offline Landbase Maps. Details on configuring these is described later in this document.

Directory Location

The GeoJSON files required by the Mobile App should be generated in the \$OPERATIONS_MODELS/export directory. The GeoJSON generator takes each *.mb file in your NMS electric model and creates a corresponding *.geojson file.

- The \$OPERATIONS_MODELS/export directory is created by the Model Build Services when the **-export** parameter is provided on the Model Build Service startup. The Model Build Service will also create an .mb file in the \$OPERATIONS_MODELS/export directory for every map built in the system. These .mb files are the inputs to the GeoJSON file generator.
- In addition to the individual GeoJSON files, a zip file (mapset.zip) containing all of the files will be created in the \$OPERATIONS_MODELS/export directory. This file will be used by the Operations Mobile Application when it is set to perform doing a bulk download of the maps.

Build Processes

Once you have the `$OPERATIONS_MODELS/export` directory created and populated with the source .mb files, you will need to create a custom script to convert your .mb files to GeoJSON files. Use the OPAL script, `OPAL-build-mobile-maps`, as a template for your script to create GeoJSON files:

1. Copy `${NMS_HOME}/OPAL/bin/OPAL-build-mobile-maps` to `${NMS_BASE}/[project]/bin/[project]-build-mobile-maps`.
2. Edit the `[project]-build-mobile-maps` script based on requirements of the GeoJSON file generation. Example changes: filter out object, skip landbase maps, cleanup data issues, and so on.
3. Add a call to the product version of the `nms-build-mobile-map` script from your `[project]-postbuild` script. The `nms-build-mobile-maps` script has a mutex scheme to prevent multiple copies of this process from running at once.

GeoJSON Configuration File

The GeoJSON generation process requires a `[project]_geojson_export.dat` file to do the following:

- Identify the classes of objects in the source .mb files to convert to GeoJSON file features.
- Identify the attributes to bring into the GeoJSON files for each object class.
- Define the coordinate conversion parameters to convert the .mb file coordinates to the mobile app required coordinate system.

Create a copy of the template (`$NMS_BASE/OPAL/sql/OPAL_geojson_export.dat`) and save it as `${NMS_HOME}/[project]/sql/[project]_geojson_export.dat`.

The header of the file has the instructions needed to configure the objects included in the GeoJSON maps.

Electrical objects in the GeoJSON files will be required to have the following attributes:

- **FeatureType:** High level feature type (for example, Electric).
- **Layer:** Layer name within the FeatureType (for example, Switch, Transformer, Fuse).
- **DeviceType:** Descriptive class of the device (for example, Three Phase OH Primary).
- **Symbol:** Symbology to apply to the device. For electric objects, add a suffix (-OPEN, -CLOSED, -MIXED) for the nominal state (for example, Transformer-CLOSED, Switch-OPEN).
- **Phase:** Phases of the device (A, B, C, AB, AC, BC, ABC).
- **NomStatus:** Nominal Status of the device (OPEN, CLOSED, MIXED).
- **NomClosedPhases:** The nominally closed phases for the device (for example, A).
- **HandleClass:** Handle Class number of the device from the Classes table in NMS (for example, 123).
- **HandleIndex:** Handle Index number of the device (for example, 1002).
- **Partition:** Partition number the device belongs to (for example, 1047).
- **DeviceId:** Alias name of the device (for example, F1461).

- **Feeder:** Feeder name the device belongs to (for example, 2414). This feeder value must match the NMS feeders table feeder_name value to allow OMA to use the color for that feeders table row.
- **Substation:** Substation name that the device or the device's feeder belongs to (for example, Lake Sub).
- **Location:** Descriptive location or address of the device.
- **IntPartitions:** An array of internal world partitions the object belongs to.

GeoJSON Map Deployment

The GeoJSON Maps consist of two types of files:

- **[mapname].geojson files:** GeoJSON versions of the NMS map files.
- **mobile_geojson_maps.json files:** The index file for the GeoJSON maps

To get the maps to the OMA client, the following methods are supported:

- At any time, you can go to the map librarian section of the OMA map page and request an updated server map index file (`mobile_geojson_maps.json`), and compare it to the already installed files, and download any outdated GeoJSON map files. OMA will also automatically check the server for updated maps on initial navigation to the map page, if new maps are available, it will inform the user.

GeoJSON Offline Landbase Maps

Overview

The Oracle Network Management System Operations Mobile App GeoJSON Offline Landbase Maps. When OMA is not connected to the Internet, the offline landbase will give the user some geographic context in the OMA map.

To minimize storage of the offline maps on the device and to optimize performance when rendering them in the map, OMA recommends utilizing street centerline data and other simple landbase features (such as railroad centerlines, water body outlines, and so on).

To configure the build process for the offline maps, you will need to configure two items for your project:

1. A `[project]-build-landbase-mobile-maps` script that knows what maps comprise the landbase maps to be processed and will generate the landbase `.geojson` maps and a map index `mobile_geojson_landbase_maps.json` file in a landbase subdirectory (`$OPERATIONS_MAPS/data/export/landbase`). The file in the landbase directory need to be copied into your OMA `src/js/landbase` directory prior to building OMA2. The OPAL model has an example `[project]-build-landbase-mobile-maps` script file in the `$NMS_BASE/OPAL/bin` directory named `OPAL-build-landbase-mobile-maps`.
2. A `[project]_landbase_geojson_export.dat` containing the definitions of the objects to transform from the landbase centerline `.mb` file to the resulting `geojson` files. OPAL has an example of this in the `$NMS_BASE/OPAL/sql` directory named `OPAL_landbase_geojson_export.dat`. It utilizes the NMS standard street centerline model plus a few OPAL specific modeling objects (railroads and waterways). Do not import any text objects or street intersection points.

The OMA product template has a set of offline landbase maps prepackaged in it covering the OPAL model area as an example in the `src/js/landbase` directory.

Mobile User Validation

Mobile user validation is done in the Network Management System Configuration Assistant and is a separate validation scheme than the Network Management System Web Workspace users.

The screenshot displays the Oracle Utilities Network Management System - Configuration Assistant interface. The top navigation bar includes the Oracle logo, the application name, the user name 'Jeanine Meier', and links for 'About' and 'Help'. Below the navigation bar, a series of tabs are visible: 'Event Details Options', 'Feeder Management', 'State Transitions', 'Default Restoration Times', 'Alarm Rules', 'Customer Administration', 'User Administration', 'User Permissions', 'Flex User Administration', 'Mobile User Administration' (which is selected), 'Mobile Applications', and 'Event Management Rules'.

The 'Mobile Users' section contains a table with the following data:

USERNAME	FIRST NAME	LAST NAME	CREATION KEY	COMPANY	PERMISSION SET	CREW	LAST CONTACT
OWA_TF	T	F		OPAL	internal		
OWA_TW	Tech	Writer		OPAL	internal		

Below the table are buttons for '+ Add...', 'Remove', 'Edit...', 'Refresh', 'Export Configuration...', and 'Login History...'.

The 'Mobile User Type Memberships' section shows two lists: 'REMAINING MOBILE USER TYPES' (Damage Assessor, Hazard Responder, View Only) and 'CURRENT MOBILE USER TYPES' (Full Operations). Buttons for '+ Add Selected Mobile User Types' and 'Remove Selected Mobile User Types' are at the bottom.

The 'Current Keys' section contains a table with the following data:

KEY	KEY GROUP	PERMISSION SET	DEFAULT USER TYPES	COMPANY	CREW TYPE	CREW PREFIX	AVAILABLE
ad_key	active directory key	internal	Full Operations, Hazar...	OPAL			
key0	storm ohio1 - IEEE	internal	Full Operations, View ...	IEEE		OMA_	1
key1	storm ohio1 - IEEE	internal	Full Operations, View ...	IEEE		OMA_	1
key10	storm ohio1 - IEEE	internal	Full Operations, View ...	IEEE		OMA_	1
key5	storm ohio1 - IEEE	internal	Full Operations, View ...	IEEE		OMA_	1

Buttons for '+ Add...', 'Remove', 'Edit...', and 'Refresh' are at the bottom.

The 'Permission Sets' section shows three lists: 'PERMISSION SET' (external, internal, service), 'AVAILABLE PERMISSIONS' (Access Events, Access Miscellaneous Log, Allow as-user Parameter, Confirm Outage, Edit Permanent Crew, Group Event), and 'CURRENT PERMISSIONS' (Allow Device Operations, Allow Switch Step Self Instruct, Allow Switch Step Updates, Change Crew, Send HLM). Buttons for '+ Add Set...', 'Remove Set', '+ Add Selected Permission', and 'Remove Selected Permission' are at the bottom.

Permissions and Permission Sets

Permissions are used to allow users to have access to functionality and information. The permissions available include:

- **Change Crew:** This access allows a user to change the crew. If it is not assigned to a user then the user will be locked to their existing crew.
- **Extended Switch Step Updates:** Client side rule to allow the user to edit more switch step fields than just the completion time and comments.
- **Allow Switch Step Updates:** Allow the user to update switch sheet transitions and update fields.
- **Allow Device Operations:** Ability to mark non-SCADA devices open or open or closed in NMS
- **Send HLM:** Allow the user to send high-level messages to the NMS.

Permission sets are groups of permissions. Users and Keys have permission sets associated to them.

Predefined Users

Mobile users can be defined in the Network Management Systems using the Configuration Assistant Mobile User Administration tab. An administrator can create the predefined username using the Mobile Users section by hitting the **Add** button and filling out the Add/Edit Mobile User Information panel and saving the changes.

Operations Mobile Application users can enter the username/password as authorized in this section to gain access to the application functionality.

Creating Users Using a Key

Mobile users may be created on the mobile client when the user is given a key authorizing the creation of a mobile user. In the mobile app login page, the user can check the **new user** box and enter in a new username, password, and the provided new user key and create a new mobile user.

The keys required to create a mobile user from the mobile application are maintained in the `mobile_new_user_keys` database table in the Network Management System. This table can be managed using the Network Management System Configuration Assistant Mobile User Administration tab in the Current Keys section. Simply add a new key with an availability count greater than zero, the key can be used by a mobile app user to create a new username/password into the system. The number of times this key can be user is based on the available number, each time the key is used, the available number is decremented.

To create a key for the mobile application, click the **Add** button at the bottom of the Network Management System Configuration Assistant Mobile User Administration Current Keys section and filling out the **Add/Edit Mobile Keys Information** panel and saving the changes.

If a key contains a crew type and crew prefix, a crew will be created automatically for the users created with the key.

For more details on the Configuration Assistant Mobile User Administration, refer to the *Oracle Utilities Network Management System User's Guide* and the *Oracle Utilities Network Management System Configuration Guide*.

Using LDAP/AD User Validation

LDAP/AD users can be configured to work with OMA. However, if you plan to support both OMA authenticated users, **Predefined** or **Keys** (as described in the previous two sections), then you need to force OMA authenticated users to have a user name prefix to eliminate potential conflicts with LDAP/AD user names. To specify a user name prefix requirement for OMA authenticated users, specify the **Crew Prefix** in the Configuration Assistant's **Mobile Users** tab's **Current Keys** section.

LDAP validation can occur in one of two places: the *nms-ws WebLogic authentication scheme* or the *cesejb WebLogic authentication scheme*. If the RESTful service calls contain an http header `nmsWsValidate: true`, then the *nms-ws WebLogic authentication scheme* will be used first prior to attempting to use the *cesejb WebLogic authentication scheme*. Otherwise, without the `nmsWsValidation` header, only the *cesejb WebLogic authentication scheme* will be used.

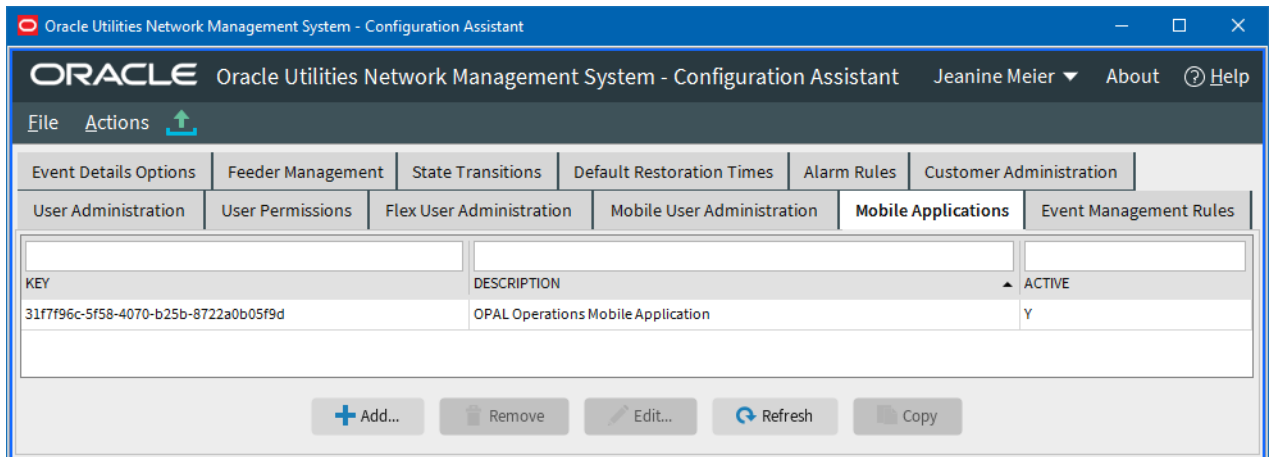
To configure LDAP/AD user validation for OMA, there are two lines that need to be enabled in the `CentricityServer.properties` file. Uncomment the last two lines, as shown below:

```
# Operations Mobile Application (OMA) user authentication
configuration
#
# If a user belongs to an ldap group starting with the defined
mobile_ldap_user_group_prefix, they are allowed access to OMA services
#
# If the value of mobile_ldap_user_group_prefix = DISABLED, no user
validation to the ldap server will be performed.
#
# If the mobile_ldap_user_group_prefix is an exact match, and the user
is new to OMA, the mobile_ldap_user_default_new_user_key is assigned
to the user.
#
# If the group_prefix is followed by -<new_user_key>, (example,
omauser-ad_key), then the <new_user_key> will be used as the
new_user_key for the new user.
#
# Once set, permissions must be changed in NMS Config Assistant for the
user.
#
# Example behavior of these new users
#
# user1 belongs to group OMAdev. This user is not given access to OMA
because the group does not start with the user group prefix
#
# user2 belongs to group omauser. This user is given access to OMA and
will have the permission defined by the default user key ad_key because
the user_group is an exact match
#
# user3 belongs to group omauser-damage. This user is given access to
OMA and will have the permission defined by the user key damage
mobile_ldap_user_group_prefix = omauser
mobile_ldap_user_default_new_user_key = ad_key
```

Further, in the `MOBILE_NEW_USER_KEYS`, the value for available should be 0 for the key "ad_key". This is the user_key that is assigned to new users through this automated process and should not be allowed to be assigned to users through any other process. Setting available = 0 will prevent this key from being assigned to users other than the new LDAP-authenticated users.

Mobile Applications Validation

Mobile Applications Validation is done in the Network Management System Configuration Assistant and is a separate validation scheme than Mobile User Validation. Mobile Application Validation consists of a application key built into the OMA application and it must match a configured model application key in the NMS Configuration Assistant Mobile Applications tab:



The **Key** must match the KEY in the OMA `src/js/resources/config/loginSettings.js` file's `self.appKey` value.

Here is the default key:

```
self.appKey = '31f7f96c-5f58-4070-b25b-8722a0b05f9d';
```

You can have as many keys in the NMS Configuration Assistant Mobile Applications tab as you like; OMA is only built with one key, but different versions of OMA can use different keys (one for internal version of the app, another for extern version of the app).

Configuring OMA Highlight Coloring

Overview

The OMA coloring for tracing, selections, and highlight coloring can be configured by modifying the section in the project's `src/js/resources/config/map.js`.

The product version is:

```
self.secondaryColor = utils.colorFromName.SLATEGREY;
self.traceColor = utils.colorFromName.CYAN;

self.lineGlow = { selectionColor: utils.colors['#ffff00'], width: 30
};

// This is the configured glow types (first one takes precedence)
// Bits to check, enabled option, color, secondary (optional.. if this
// is not set then the feature is used for both secondary and primary)

self.lineGlowTypes = [
    [utils.energizationState.ASSESSED_BIT, 'assessed',
    utils.colors[utils.colorFromName.ORANGE]],
    [utils.energizationState.IN_OUTAGE_BIT |
    utils.energizationState.DEENERGIZED_BIT, 'confirmed_deenergized',
    utils.colors[utils.colorFromName.BLACK]],
    [utils.energizationState.IN_OUTAGE_BIT |
    utils.energizationState.DEGRADED_BIT, 'confirmed_deenergized',
    utils.colors[utils.colorFromName.BLACK]],
    [utils.energizationState.IN_OUTAGE_BIT, 'predicted_deenergized',
    utils.colors[utils.colorFromName.DEEPPINK]]
```

secondaryColor: If this is defined then devices that are part of the secondary network will be set to this color, instead of following feeder or phase coloring.

traceColor: The color to display traces

lineGlow: The line style to use when displaying conductor highlighting

lineGlowTypes: The colors to use for assessed, deenergized, and predicted outages.

Server Based Documents

Overview

The OMA **Documents** panel allows you to access standard corporate documents. These documents can be updated from the NMS server so latest versions are available. Examples of documents that might be useful to store in NMS include standard procedures and safety documents; file types should be limited to types that OMA devices would most likely support, such as PDF, DOC, and so on.

Document Locations

Documents that you want to be available to OMA should be stored on your NMS server in the `$(OPERATIONS_MODELS)/docs` directory.

Document Configuration in OMA

In the OMA2 project `src/js/resources/config/documents.js` file, add a record in the `available_documents` object for each required document. Here is an example for two documents:

```
self.available_documents = [  
    {name:'OPAL_safety_rules.pdf', description:'Safety Rules  
(PDF)'},  
    {name:'OPAL_safety_rules.doc', description:'Safety Rules (DOC)'}  
];
```

Configuring OMA Object Attribute Viewer

Overview

The OMA map selection attribute viewer (**Attributes** dialog box) allows you to view a selected object's attributes. By default, all of the object's attributes contained in the GeoJSON file will be shown. The GeoJSON file can be configured to include attributes on objects based on object class (see [GeoJSON Configuration File on page 4-2](#)). You can configure the presentation of the attributes in the **Attribute** dialog box; for example, you may set which attributes to display, their attribute names, and the order presented. The **Show Source Attributes** option on the **Attributes** dialog box allows you to see all of the attributes on an object, which may be needed for support or debugging.

Configuration

The attribute viewer configuration is defined in the OMA project's `src/js/resources/config/map.js` file where you can define the attribute labels and the order of objects. If a layout is not defined for a object, the default is to show all attributes on the object.

Use this definition to describe the attribute table layout:

```
self.tableViewerConfig = [
  {
    matchLayerNames: [<layer names to match> ],
    matchAttributes: [ {name:<match-attribute-name>,values:[
<match-attribute-values>, .... ]},... ],
    attributes: [
      {attributeName: <attribute-name>, attributeLabel: <attribute-
label>}, ....
    ]
  }, ....
]
```

- `<layer names to match>` will come from the `feature.feature.feature_layer.name` value from the map object.
- `<match-attribute-name>` will come from the `feature.feature.attributes` list and the value must match one of the `<match-attribute-values>`.
- If no `matchingAttributes` listed, it will be considered a match.
- If more than one `matchingAttributes` are given, they all must match.
- Available condition attributes can be configured using a view with the `"_oma_view"` suffix. For example, create a view called `HOLDS_OMA_VIEW` on the `HOLDS` table and join safety documents, sheet information, and device attributes as desired.
- `tableViewerConfigs` will be processed in order, and the first match will be used.

Example

```
self.tableViewerConfig = [
    {
        matchingLayerNames: [
            "Conditions"
        ],
        matchingAttributes:
        [{name:"LINK_TYPE",values:["associated_document"]}],
        attributes: [
            { attributeName: "LINK_TYPE", attributeLabel: "Condition
Type" },
            { attributeName: "TEXT", attributeLabel: "File Name" },
            { attributeName: "TIME", attributeLabel: "Date/Time" },
            { attributeName: "OBJECT", attributeLabel: "Device" }
        ]
    }
]
```

Configuring OMA For Schematic Maps

Overview

OMA can support viewing NMS generated schematic maps, including general schematics and single circuit schematics. Schematic maps can double or triple the volume of OMA map data being generated on the server and being managed on the OMA device. Due to the performance implications, it is advisable to do a business case justification to determine if schematics are warranted, and, if so, what schematic sets to include in OMA.

Configuration

To configure NMS and OMA to support schematics, please complete the following:

- Change your `[project]_gejson_export.dat` file to include `ALT_DIAGRAM`, `LATITUDE`, and `LONGITUDE` attributes on objects likely to be included in schematics to allow OMA to navigate from the geographic and internal maps to the schematic maps. Typically, object to include are primary switches, breakers, and 3 phase conductors and cables.
- Change your `[project]_gejson_export.dat` file to include `.mb` objects that are new in the schematic maps. Typically this would include substation boxes, switch gear boxes, and schematic connectors. Check your `<project>-create-schematics` script for schematic specific classes.
- Change your `[project]_gejson_export.dat` file to include a record to define which of the schematic sets are single circuit verses general schematics. Single circuit schematics will have a special coordinate system assigned to them in the `<project>-create-schematics` script `-coordsystem` parameter on the `schematica` call for single circuit schematics. This value is also in the `VIEWER_GLOBAL_PROPERTIES.inc` file `viewer.single_circuit_schematic_coord_sys` property (defaults to 10 for product).
- Change your `[project]-create-schematics` script and add the `-export` flag to any `schematica` calls for schematic map sets you want included in OMA. This flag will tell the `schematica` program to write out `.mb` files to the `$OPERATIONS_MODELS/export` directory to be processed by the `gejson` generation process.
- Change your model build post-process script (`[project]-postbuild`) to do schematic map generation before the OMA `gejson` map generation.
- Change your `[project]-build-mobile-maps` script to process `schematica .mb` files in the `$OPERATIONS_MODELS/export` directory.
- Change your project version of partition based condition views to include `dev_cls` and `dev_idx` fields. For product, these views include `oma_truck_locations_ptn`, `oma_incidents_ptn`, `oma_jobs_ptn`, and `oma_damage_reports_ptn` and are defined in the `[project]_schema_mobile_crew.sql` file.

Configuring OMA Search Options

Overview

OMA search options can be project configured to search for objects in the NMS model. The OMA source code provided is set up to search the OPAL model and may need to be tweaked in order to work with a specific project model. In general, the predefined searches for Device ID, Customer information (name, phone, address, account ID, and meter ID), Feeder Name, and Latitude and Longitude should work as is for most projects however the Substation search (dependent on use of class 10210/substation fences), Site ID search, and Street Intersection search will most likely need to be changed to match project models.

Configuration

The configuration of the OMS search is in three places:

- [NMS Database mobile_search Views](#)
- [OMA src/js/resources/config/map.js self.searchDefinitions](#)
- [OMA src/js/viewModels/map.js and src/js/views/map.html](#)

NMS Database mobile_search Views

Add views in the database to support OMA searches. By convention, OMA search view names should start with `mobile_search_` (for example, `mobile_search_device_view`). Be sure to add them to the `GET:/mobile/dataset/{table}` API whitelist table (`mobile_dataset_tables`) and to the `READ_ONLY_TABLES`. OPAL defines these in the `OPAL_schema_mobile_crew_setup.sql` file, which is called by the `OPAL-mobile-crew-setup` script. Projects should define their own versions of these files.

OMA search views should always contain these fields:

- **DEV_CLS:** object class number of search objects
- **DEV_IDX:** object index number of search objects
- **<search field>:** any field you use as a search value (for example, device id)

You can include the following optional fields:

- **X_COORD and Y_COORD:** if these are provided, the `GET:/mobile/dataset/{table}` API may use these to identify the location to focus on (see the API description below).
- **LATITUDE and LONGITUDE:** if these are provided, the `GET:/mobile/dataset/{table}` API may use these to identify the location to focus on (see the API description below).
- **<more fields>:** any other fields to provide context to the OMA user if multiple records match the user's search criteria (such as location, feeder, city, and so on).

Example

The following example is from the OPAL_schema_mobile_crew_setup.sql file:

```
create or replace view mobile_search_device_view as
select a.alias "DEVICE_ID",
c.c_desc "DEV_TYPE",
a.h_cls "DEV_CLS",
a.h_idx "DEV_IDX",
(select coalesce((
  select cz.name from network_components nc, control_zones cz
  where a.h_cls = nc.h_cls and a.h_idx = nc.h_idx and nc.active = 'Y'
  and cz.death is null and nc.ncg = cz.ncg_id
), 'NONE') from dual) "FEEDER",
(select coalesce((
  select cz2.name from network_components nc, control_zones cz,
Control_Zone_Structures czs, control_zones cz2
  where a.h_cls = nc.h_cls and a.h_idx = nc.h_idx and nc.active = 'Y'
  and cz2.death is null and cz.death is null and czs.active = 'Y' and
nc.ncg = cz.ncg_id
  and cz.ncg_id = czs.child_ncg_id and cz2.ncg_id = czs.parent_ncg_id
), 'NONE') from dual) "SUBSTATION",
(select coalesce((
  select pc.x_coord from point_coordinates pc, partitions p
  where a.h_cls = pc.h_cls and a.h_idx = pc.h_idx and pc.active = 'Y'
  and p.active = 'Y'
  and pc.partition = p.h_idx and p.coord_system = 0 and pc.sequence = 1
  and p.h_cls = 4001
), NULL) from dual) "X_COORD",
(select coalesce((
  select pc.y_coord from point_coordinates pc, partitions p
  where a.h_cls = pc.h_cls and a.h_idx = pc.h_idx and pc.active = 'Y'
  and p.active = 'Y'
  and pc.partition = p.h_idx and p.coord_system = 0 and pc.sequence = 1
  and p.h_cls = 4001
), NULL) from dual) "Y_COORD"
from alias_mapping a,
classes c
where a.active = 'Y'
and a.db_type = 'OPS'
and a.h_cls = c.c_numb
and upper(c.c_type) like 'ATT\_%' ESCAPE '\'
and upper(c.c_type) not like '%STREET%'
and upper(c.c_type) != 'ATT_SUPPORT_STRUCTURE'
;
grant SELECT, INSERT, UPDATE, DELETE ON mobile_search_device_view TO
CES_RW;
GRANT select ON mobile_search_device_view TO CES_R0;
delete from mobile_dataset_tables where OBJECT_NAME =
'mobile_search_device_view';
insert into mobile_dataset_tables (OBJECT_NAME, ACCESS_LEVEL) values
('mobile_search_device_view', 'READ');
delete from READ_ONLY_TABLES where OBJECT_NAME =
'mobile_search_device_view';
insert into READ_ONLY_TABLES (OBJECT_NAME) values
('mobile_search_device_view');
COMMIT WORK;
```

OMA src/js/resources/config/map.js self.searchDefinitions

The format for the search definitions is:

```
self.searchDefinitions = [
  {
    searchType: "Device ID",
    field1Label: "Device ID:", field1Type: "String",
    field2Label: "", field2Type: "",
    searchTable: "mobile_search_device_view", searchColumn:
    "device_id", needs_location: 'Yes',
    note: "Use % for multi-character wild card.<br>Use _ for
    single-character wild card.",
    multiSelectAtts: [
      {fieldName: "device_id", fieldLabel: "Device ID"},
      {fieldName: "feeder", fieldLabel: "Feeder"},
      {fieldName: "substation", fieldLabel: "Substation"},
      {fieldName: "dev_type", fieldLabel: "Device Type"}
    ],
    multiSelectTitle: "Device ID Search Results",
    showAttsOnOneResult: false
  },
  ...
]
```

All fields are required (unless a default value is noted):

- **searchType:** name used on the top of the search panels and on the search type pull-down on the search panel.
- **field1Label:** label to use for search filed one on the search dialog panel.
- **field1Type:** use to validate the field entry, can be on of these values: String, Number, Phone, or COORD.
- **field2Label and field2Type:** used in search dialog if field2Label is not "".
- **searchTable:** the mobile_search database table or view to use for the search
- **searchColumn:** column name to use in the search table. If using two fields on the search panel, the two field values will be combined with a " & " (%20%26%20) between them.
- **needs_location:** Yes or No. If yes, the RESTful API will get a lat-long=true, which will cause the API to attempt to put _LATITUDE, _LONGITUDE, and LOCATION attributes on the result set.
- **note:** any special instructions you would like included on the search panel (for example, wild card instructions, format details, and so forth).
- **multiSelectAtts:** A list of fields to display to the user if they need to select a specific search result. The objects in the list will have two components:
 - **fieldName:** name of the result seat field.
 - **fieldLabel:** Label to use for the field.

These will be displayed in order in a tabular form in a multiple result set panel for the user to select from.

- **multiSetlectTitle:** Label to put on the top of the multiple result set panel.
- **showAttsOnOneResult:** True or False. If true, the search results will always be shown to the user, even if only one record is returned for the search. If false, a single record result set will automatically focus the map without presenting the result set to the user.

- **focusPriority:** {GEO|INT|SUM} - Default is GEO. GEO will focus on the geographic location if the object has a geographic representation, otherwise it will next focus on an internal location if it has one, finally we will focus on the summary location. INT will focus on the internal location if the object has an internal representation, otherwise it will focus on the geographic location. SUM will always focus on the summary location if non-zero, otherwise it will try to focus on the internal location.
- **intFocus:** {COORD-N|PARTITION} - Default is COORD-6. intFocus will specify the type of focus to do if the target map is an internal map. COORD-N will center the map on the search object coordinate and do N zooms. PARTITION will leave the map to view the entire partition.
- **offlineDataFile:** "filename" - If not specified, default is offline search is not configured for this search definition. If specified, OMA will download the specified file from the NMS Server \$OPERATIONS_MODELS/offline_search directory. File must be in JSON format and match the data structure of the RestFUL service response. OPAL has an example script to build the offline search JSON files (OPAL-build-mobile-offline-search-files).

You can include as many search definitions in the array as needed.

There is one special search definition (Latitude and Longitude) in the OPAL/OMA configuration that does not make any service calls and directly focuses the map:

```
{
  searchType: "Latitude and Longitude",
  field1Label: "Latitude,Longitude:", field1Type: "Coord",
  field2Label: "", field2Type: "",
  searchTable: "", searchColumn: "",needs_location: 'No',
  note:"Please enter both Latitude,Longitude<br>separated by
  space or comma.",
  multiSelectAtts: [],
  multiSelectTitle: "",
  showAttsOnOneResult: false
}
```

OMA src/js/viewModels/map.js and src/js/views/map.html

map.js and map.html may need changes based on special search actions you may want to perform.

The search function (map.js: self.doSearchClick) provides special actions based on the search type (self.mapSearchType). An example is the Latitude and Longitude search where we don't need to query the NMS system and it can be done while offline; whereas the other searches all require online access to the NMS server.

The map.html file contains the search dialog box definitions (MapSearchPopup and MapSearchSelectPopup). Based on special processing you might need for search options, these can be changed. An example of this is the Latitude and Longitude search where the wild card options are hidden since it is not relevant to this type of search.

Chapter 5

Client Development Setup

This chapter describes installing, building, and testing the Operations Mobile Application. You can set up the client development environment on Windows 10, MacOS, or Linux. Please refer to the Oracle JET (JavaScript Extension Toolkit) documentation for further details.

- [Install Software](#)
- [Build Operations Mobile Application](#)
- [Testing](#)
- [OMA Client Configuration](#)

Install Software

Install Prerequisite Software

Install the Prerequisite Software as defined in the Supported Platforms and Hardware Requirements Chapter Prerequisite Software Section of this document.

If you are targeting the Android platform for the application, install the Android SDK. If you are targeting the iOS devices, install XCode and Apple Developer ID for iOS. If you are targeting the Windows 10 or newer platform, install the Microsoft Visual Studio.

You should set the following to point to your Android Home location:

```
ANDROID_HOME=/Users/appbuild/Library/Android/sdk
```

Install Operations Mobile App SDK

The Operations Mobile App SDK is located in the `$NMS_DIST/sdk/OMA2.zip` and `$NMS_BASE/sdk/OMA2_3rdparty.zip` files of your Oracle Network Management System. Copy these zip files to your development build environment system and unzip them. This will be your Oracle JET project directory.

Build Operations Mobile Application

To build the OPERATIONS MOBILE APPLICATION, we have provided a template script in the install package.

Using a terminal window, change to your Oracle JET project directory where you unzipped OMA2.zip:

```
cd /Users/appbuild/OMA2
```

Look for the `oma2_build.sh` script.

Execute this script in order to build OMA. Here is the format of the command:

```
./oma2_build.sh [-help] [-windows] [-ios] [-android] [-browser]
```

Also you may add the following arguments:

- `--proxy_url` - the URL for your proxy server (if required)
- `--proxy_port` - the port number for your proxy server (if required)
- `--registry_url` - The URL for an internal npm registry (if required)
- `--ios_keychain_path` - The path to the iOS login keychain (required for -ios)

Testing

Test with a Browser

To test with a browser, run these commands:

```
$ /users/appbuild/OMA2 # your OMA 2 build directory
$ ojet server ios --browser=firefox # can be one of these
[chrome|firefox|edge|ie|safari]
```

This will bring up an http server serving up the OMA application. It will also start OMA in the specified browser.

Test with iOS Device

You can test the application using iOS devices using these methods:

- XCode iOS Emulators
- XCode Debug Installer using an iPad or iPhone and a USB Cable
- Install the ipa file using iTunes or iFunBox.

Test with Android Device

You can test the application using Android devices using these methods:

- Android SDK Emulators
- Android SDK Installer using an Android Device and a USB Cable
- Android .apk installation directly on an Android Device

Test with Windows 10

You can install the application package (.appx file) using the Desktop App Installer on any PC or Windows 10 device.

OMA Client Configuration

OMA client has a number of configuration file projects should expect to adjust for project needs. Most of the files are in the `~/src/js/resources` and `~/src/js/resources/config` directories. Please review these directories for any project configuration settings you might like to change.

In addition to the previously described configuration areas, there are some configuration topics to review based on your project needs.

Third Party Application Integration with the OMA Client

OMA supports two different methods to allow third party integration:

- Custom URI integration for Android and iOS OMA clients.
- HTTP Bridge for Windows 10 OMA clients and browser testing clients on a Windows 10 device

Third Party Application Integration with Custom URI (Android and iOS)

The OMA Android and iOS client applications support a custom URI integration scheme where other applications running on the OMA device can interact with the OMA application. This feature has been implemented using the Cordova plugin named `cordova-plugin-customurlscheme`. For further details, please refer to the Cordova Plugin Documentation online.

OMA will need to be configured to process Custom URI commands in the `oma-main.js::handleOpenURL` function. OMA has some preconfigured custom URIs prebuilt into the sample OMA application that support the API command URIs:

1. Focus map on latitude/longitude coordinates

```
nmsoma://mapfocus?lat=123&long=456&msg=message
```

OMA map page will open and focus on the latitude and longitude coordinates supplied by the **lat** and **long** parameters. Message passed in the **msg** parameter will be displayed to the user. If no message is passed, then an internally configured message will be displayed. Optional parameter, **zoom**, allows you to specify zoom level.

2. Update crew status

```
nmsoms://crewstatus?status=ONS&event=123
```

Status of the crew associated with the OMA user will be set to the value specified by the **status** parameter (valid values: ASN, ENR, ONS, SUS, CNL, INC) with respect to the event specified by the **event** parameter. Crew status will only be updated in OMA; it will not be sent to NMS. The statuses that OMA will allow another application to use must be configured in the `src/js/resources/config/tasks.js` file in the following value:

```
// crew event statuses that are allowed to be set by external app
self.allowed_external_task_statuses = ['ASN', 'ENR', 'ONS', 'SUS',
    'CNL', 'INC'];
```

3. Display specific OMA screen

```
nmsoma://open?page=switching&id=123
```

Opens OMA screen specified by the **page** parameter (valid values: `switching`, `task`). If the **id** parameter is provided, then details for the specific switching sheet or task will be displayed.

4. Return data field

```
nmsoma://getinfo?type=event&id=123&responseurl=otherapp://  
data?cause={cause}
```

This request allows other applications to retrieve information from OMA. Parameter **type** specifies the type of the object from where information will be retrieved (valid values: event). Parameter **id** specifies the object identifier. Parameter **responseurl** contains template for the custom URL that will be used to deliver response to the other app. Template parameters enclosed in **{ }** will be substituted with data values.

If OMA is requested to navigate to a different screen when there are unsaved changes, the user will be prompted to save the changes, discard them, or ignore the navigation request.

Third Party Application Integration with localhost HTTP Bridge (Windows 10)

The OMA Windows 10 client and browser test client applications support a localhost HTTP bridge URL integration scheme where other applications running on the OMA device can interact with the OMA application. This feature has been implemented using an HTTP server installed on the windows device running windows service that contains a server listening for requests from 3rd party apps and allowing OMA to pick up the request and process them.

On the Windows machine, you will need to install the OMA Bridge Service from the NMS SDK directory `OMAHttPBridge.zip` file. By default, the bridge app will start every time the system is rebooted; the user can use the Windows Service app to monitor and control (Start/Stop) the OMA Bridge Services. The Windows Event Viewer can also be used to view the log activity of the OMA Bridge in the Event Viewer, Applications and Services Logs, `OMABrdg` section.

The default port for the `OMAHttPBridge` is 9595. To change the port number, you will need to manually start the service with a single parameter: the desired port number.

Calls from the third party app would be as follows:

```
http://localhost:9595/oma/<command>
```

Where *<command>* would be some action OMA is prepared to do. For example:

```
http://localhost:9595/oma/mapfocus?lat=45.1272&long=-  
93.5023&zoom=12
```

Will return a json payload including the following information:

```
{  
  "omaStatus": "LoggedIn",  
  "omaRequest": "/oma/mapfocus?lat=45.1246&long=-93.5023&zoom=18",  
  "requestStatus": "queued"  
}
```

The payload **omaStatus** will return `LoggedIn`, `LoggedOut`, or `unknown`. If OMA is not running or connected to the `OMAHttPBridge`, **omaStatus** will be `unknown` and OMA will not respond to the request. If OMA is connected to the bridge but not logged in, **omaStatus** will be `LoggedOut` and OMA will most likely not respond to the request.

Other commands supported by the OMAHttpBridge other than the `http://localhost:9595/oma/` commands include:

`http://localhost:9595/` - Which will return a status of the OMAHttpBridge:

```
{
  "appName": "OMAHttpBridge",
  "version": "16FEB2021-A",
  "omaStatus": "LoggedIn",
  "omaRequest": "/oma/mapfocus?lat=45.1269&long=-93.4994&zoom=18",
  "requestStatus": "none",
  "totalRequests": 1083,
  "omaRequests": 15,
  "omaGetRequests": 1067
}
```

`http://localhost:9595/getrequest:` Which OMA uses to get the last requested oma command:

```
"omaRequest": "/oma/mapfocus?lat=45.1272&long=-93.5023&zoom=18"
}
```

`http://localhost:9595/notloggedin:` Which OMA uses to signal the user is not logged in to OMA.

`http://localhost:9595/stop:` Which can be used to terminate the OMAHttpBridge.

OMA will need to be configured to process `/oma/` commands in the `oma-main.js::handleOpenURL` function just as the Custom URI method used for Android and iOS devices; however, the invocation of these command will be directed to the Http Bridge as follows:

`http://localhost:9595/oma/mapfocus?lat=123&long=456&msg=message`

`http://localhost:9595/oma/crewstatus?status=ONS&event=123`

`http://localhost:9595/oma/open?page=switching&id=123`

`http://localhost:9595/oma/getinfo?type=event&id=123&responseurl=otherapp://data?cause={cause}`

Refer to the [Third Party Application Integration with Custom URI \(Android and iOS\)](#) on page 5-3 for detailed descriptions of the above commands.

Chapter 6

Client Deployment

The deployment of the Oracle Network Management Operations Mobile Application is completely up to the Utility's IT department. This chapter will identify options to consider based on the deployment platform.

- Android
- iOS
- Windows

Android

Android platforms have these options for deployment:

Google Play Store

The Google Play Store provides public access to your Android application. See Google's developers website for details: <http://developer.android.com/distribute>

Alternative Distribution Options

The Google offers options to distribute your application through any App Marketplace, Email, or your private or public website. In order to install an app on your device that does not originate from the Google Play Store, the device will need to set the "Opt-In for Apps from Unknown Sources." See Google's developer's website for details: <http://developer.android.com/distribute/tools/open-distribution.html>

Pre-Installed Devices

You could make Android devices available for your mobile users (both internal and external) where you pre-install the Operations Mobile Application on device.

IT Installation Service

You could provide a service by your IT team to install the Operations Mobile Application on internal or external users own devices.

iOS

Apple supports the following methods to distribute your application. It is up to your IT department to determine the best deployment strategy for your iOS application.

App Store

Apple provides public access to your iOS application. See the Apple iOS developer website for details.

iOS Developer Enterprise Program

The iOS Enterprise Distribution program allows a company to distribute their own in-house apps directly. It is intended for employees of the licensee company only and that licensee must be a company or organization with a DUNS number.

Custom B2B Apps Program

Apple has programs for volume purchasing and custom B2B apps. These programs operate from the online Business Store. The Volume Purchasing Program allows businesses to buy apps from the public App Store in bulk. Custom B2B Apps extend the Volume Purchase Program for custom B2B apps built by third-party developers. The third-party developer determines which Volume Purchase customer(s) can purchase a given app. Such apps are not available on the public App Store but only through the Business Store.

Ad Hoc Distribution (intended for Testing)

Ad Hoc Distribution allows you to distribute apps to up to 100 iOS devices for testing. You must register these devices manually by their ID. Devices can be removed/replaced once each membership year). Ad Hoc Distribution is a feature of both the iOS Developer Program and the iOS Developer Enterprise Program.

iOS Beta Testing Service: TestFlight

TestFlight is a free over-the-air platform used to distribute beta and internal iOS applications to team members. Developers can manage testing and receive feedback from their team with TestFlight's Dashboard.

TestFlight makes use of your iOS Enterprise License or Developer License to create Enterprise and Ad Hoc provisioned apps.

Windows

Windows platforms have these options for deployment:

Windows Store

The Windows Store provides public access to your Windows application. See Microsoft's developers website for details: <https://developer.microsoft.com/en-us/store/publish-apps>.

Alternative Distribution Options

Installation of a Windows app (using an .appx file that you build with Cordova) can be installed on Windows 10 version 1607 or newer using the App Installer. Install by double clicking the .appx file and following the instructions.

Note: see the Microsoft article on the process for more information:
<https://blogs.msdn.microsoft.com/appinstaller/2016/05/27/app-installer/>

Pre-Installed Devices

You could make Windows tablets or PC available for your mobile users (both internal and external) where you pre-install the Operations Mobile Application on device.

IT Installation Service

You could provide a service by your IT team to install the Operations Mobile Application on internal or external users own devices.

Chapter 7

Operations Mobile Application Setup on OPAL

The OPAL demonstration project has been configured to be Operations Mobile Application ready. Below are the steps to complete to get the Operations Mobile Application running on an existing running OPAL system:

1. Change the MBSservice startup parameters to include the `-export` option. This can be done in the `$NMS_HOME/etc/system.dat` file:

```
program MBSservice      MBSservice      -dbname mb -export
```

If that parameter was already on your MBSservice startup, skip to step 3.

2. Restart MBSservice with net parameter:

```
$ Action any.MBSservice stop
$ sms-start -f system.dat
```

3. Generate new .geojson map files:

```
$ for f in `DBQuery "select filename from partitions where active
= 'Y' and filename like '%.mad' and coord_system = 0;"`
do
DiagramBuilder -map ${f%.mad}
done
$ OPAL-build-mobile-maps
```

4. Setup an Operations Mobile Application development environment on OSX, Windows 10, or Linux as describe in this document.
5. Copy the OPAL geojson map .zip file (geojson_maps.zip) and index file (mobile_geojson_maps.json) to your Operations Mobile Application development environment `nms_crew/www/data` directory. For information on the process to create these files, see [GeoJSON Map Deployment on page 4-3](#).
6. Configure the default NMS Mobile Gateway URL into the development environment file `src/js/resources/config/loginSettings.js` file by changing these lines using your server and port:

```
// Specify server configurations you would like to have
automatically configured for OMA (up to 9);
self.server1 = {
  name: 'Production',
  host: '[servername].[domain].com',
  app: 'nms-ws',
  port: 7102,
};
```

7. Build the Operations Mobile Application install file for each targeted platform/device.
8. Install the Operations Model Application on each targeted device.
9. Deploy the Mobile Gateway on the WebLogic Application Server.
10. Test the Operations Mobile Application.

Chapter 8

Operations Mobile Application Project Setup

To configure the Operations Mobile Application on an existing NMS system, follow these steps:

1. Change the MBSservice startup parameters to include the `-export` option. This can be done in the `$NMS_HOME/etc/system.dat` file:

```
program MBSservice MBSservice -dbname mb -export
```

If that parameter was already on your MBSservice startup, skip ahead to step 3.

2. Restart MBSservice with net parameter:

```
$ Action any.MBSservice stop
$ sms-start -f system.dat
```

3. Configure the Operations Mobile Application GeoJSON map generation process as defined in this document.

4. Generate new .geojson map files:

```
$ for f in `DBQuery "select filename from partitions where active
    = 'Y' and filename like '%.mad';"`
do
    DiagramBuilder -map ${f%.mad}
done
$ [project]-build-mobile-maps
```

5. Setup an Operations Mobile Application development environment on OSX, Windows 10, or Linux as describe in this document.
6. Copy the geojson map zip file (`geojson_maps.zip`) and index file (`mobile_geojson_maps.json`) to your Operations Mobile Application development environment `nms_crew/www/data` directory. For information on the process to create these files, see [GeoJSON Map Deployment on page 4-3](#).
7. In the development environment, configure the default NMS Mobile Gateway URL into the development environment file `src/js/resources/config/loginSettings.js` file by changing these lines using your server and port:

```
// Specify server configurations you would like to have
automatically configured for OMA (up to 9);
self.server1 = {
    name: 'Production',
    host: '[servername].[domain].com',
    app: 'nms-ws',
    port: 7102,
};
```

8. In the development environment, configure your device symbology. Put the device .svg files in the `src/js/symbols` directory. Map the files to the geojson object `SYMBOL` attribute in the `src/js/resources/config/map.js` file using the `self.deviceSymbolDefs` object function.

-
9. In the development environment, configure your conductor symbology. Map the symbology to the geojson object SYMBOL attribute in the `src/js/resources/config/map.js` file using the `self.electricLineSymbolDefs`, `self.abnormalLineSymbolDefs`, `self.lineGlowSymbolDefs`, `self.conductorFlowDirectionSymbolDefs` objects.
 10. In the development environment, configure your condition symbology. Put the device .svg files in the `src/js/symbols` directory. Map the files to the condition class number and condition status number attributes in the `src/js/resources/config/map.js` file using the `self.conditionSymbolDefs` objects.
 11. In the development environment, configure the classes of interest to get from the server. In the `src/js/api/mobile-api.js` file, locate the functions `self.getConditionsByLatLong` and `self.getConditionsByPartition` and adjust the list of condition types you want to use. The line will look like this:

```
let urlAppendage = "/mobile/conditions?qt=event,da,truck,instruct,incident,assessment,note,tag,info,clear,hold,hot,disable,wire_down,warn,ground,dcz,associated_document";
```

12. In the development environment, configure the `self.event_type_labels` in the `src/js/resources/config/tasks.js` file to match your project:

```
// labels for NMS event types
self.event_type_labels = {
    NO_OUTAGE: 'Fuzzy Event',
    PROBABLE_SERVICE_OUTAGE: 'Probable Service Outage',
    PROBABLE_DEVICE_OUTAGE: 'Probable Device Outage',
    ...
}
```

13. In the development environment, configure the `customer_type_labels` in the `src/js/resources/config/tasks.js` file to match your project:

```
// labels for NMS critical customer types (From
JBotFormat_en_US.properties file)
self.customer_type_labels = {
    0: 'Standard',
    1: 'Emergency',
    2: 'Key',
    3: 'Medical'
    ...
}
```

14. In the development environment, configure the `picklist_type` in the `src/js/resources/config/tasks.js` file to match your project:

```
// GUI types
// matches PICK_ENV_MAPPING in JBotFormat_en_US.properties
self.picklist_type = {
    NO_OUTAGE: 'radial',
    PROBABLE_SERVICE_OUTAGE: 'radial',
    PROBABLE_DEVICE_OUTAGE: 'radial',
    REAL_SERVICE_OUTAGE: 'radial',
    ...
}
```

-
15. In the development environment, configure the `picklist_filters` in the `src/js/resources/config/tasks.js` file to match your project:

```
// Event Details filters for each outage type
self.picklist_filters = {
  'radial':
  [
    {
      matches: [
        {
          option: "system_om",
          option_values: ["Distribution OH"]
        }
      ],
    },
    ...
  ],
}
```

16. In the development environment, configure the `picklist_cfg` in the `src/js/resources/config/tasks.js` file to match your project:

```
/ label matches FIELD_NAME in MessageCode_en_US.properties
// if visible is true then corresponding list will be
// displayed
// if required is true then non-default value is required
// to complete event (making field required implicitly
// makes it visible)
self.picklist_cfg = {
  system_om: { label: 'System', visible: true, required: false
},
  cause_om: { label: 'Sub-System', visible: true, required:
false },
  type_om: { label: 'Type', visible: true, required: false },
  ...
}
```

17. In the development environment, configure the `enableRootedOverride`, `appKey`, `projectVersion`, and `productVersion` in the `src/js/resources/config/loginSettings.js` file to match your project, keeping the formatting the same as the template:

```
self.enableRootedOverride = true;
self.appKey = '31f7f96c-5f58-4070-b25b-8722a0b05f9d';
self.projectVersion = "OPAL 2.5.0.0.0 Rev 1";
self.productVersion = "OMA 2.5.0.0.0";
```

18. In the development environment, configure your map object attribute viewer options as defined in [Configuring OMA Object Attribute Viewer on page 4-10](#).

19. In the development environment, configure the `section_options` in the `src/js/resources/config/damage.js` file to match your project:

```
self.section_options = [
  {value: 'Service', label: 'Service'},
  {value: 'Secondary', label: 'Secondary'},
  {value: 'Lateral', label: 'Lateral'},
  {value: 'Backbone', label: 'Backbone'},
];
```

20. In the development environment, configure your map object Search options as defined in [Configuring OMA Search Options on page 4-13](#).

-
21. In the development environment, configure the `location_options` in the `src/js/resources/config/damage.js` file to match your project:

```
// location
self.location_options = [
    {value: 'Street', label: 'Street'},
    {value: 'Rear Lot Line', label: 'Rear Lot Line'},
    {value: 'Other', label: 'Other'},
];
```

22. In the development environment, build the Operations Mobile Application install file for each targeted platform/device.
23. Configure your Mobile User Validation including predefined users, new user keys, and/or LDAP/AD authentication in the NMS Configuration Assistant or project scripts or `jconfig`.
24. Configure your Mobile Application Keys in the NMS Configuration Assistant or project scripts.
25. Install the Operations Model Application on each targeted device.
26. Deploy the Mobile Gateway on the WebLogic Application Server.
27. Test the Operations Mobile Application.

Appendix A

Restricted Use and User License Terms

Mobile Archive Restricted Use

The Oracle Utilities Network Management System Program includes one or more mobile application archives or libraries (each a “Mobile Archive”). Your use of the Mobile Archive is limited to the following:

1. Modify the Mobile Archive to include your custom branding, look and feel, and functionally extensions;
2. Insert your brand or logo where indicated (removing Oracle’s brands, logos, and trademarks, if any, but not removing or modifying any Oracle copyright statements except as stated in the following paragraph) in the Mobile Archive;
3. If you modify the Mobile Archive as set forth above, append the word “Portions” before any Oracle copyright statement (as an example, “Portions Copyright © 2015, Oracle and/or its affiliates. All rights reserved.”)
4. Compile, complete, and sign the Mobile Archive with your own mobile operating system-specific certificate(s), thereby creating a mobile application (“Mobile Application”); and
5. Distribute the Mobile Application within your enterprise or entity to your internal users and/or to your third party end users (“End Users”). You may not distribute the Mobile Archive to your internal end users except to the extent necessary for the creation of the Mobile Application. You may not distribute the Mobile Archive to End Users.

With respect to your distribution of the Mobile Archive as included in a Mobile Application (a) you must abide by the terms and conditions in the Programs license agreement pertaining to separately licensed third party technology and the separate terms applying to such technology, and (b) these terms constitute your order under which you are permitted to distribute the Mobile Archive portion of the Programs. With respect to creating a Mobile Application, you acknowledge that you must separately agree to and abide by license terms with the applicable mobile operating system provider and possibly other third parties. For example, for iOS applications, you agree that the Mobile Application, in whole or in part, may not be installed on a mobile device or executed except as incorporated into an iOS application that has been signed using an appropriate Apple-issued certificate that you obtained directly from Apple and that is deployed in full compliance with your agreement with Oracle (including these terms) and license terms set forth in a separate agreement between you and Apple.

