Oracle Utilities Network Management System

Operations Mobile Application

Installation and Deployment Guide

Release 2.4.0.0.0

E98857-03

December 2018

Revised January 29, 2019

ORACLE

Oracle Utilities Network Management System Operations Mobile Application Installation and Deployment Guide, Release 2.4.0.0.0

E98857-03

Contents

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 Release 2.3.0.1 documentation set:

- *Oracle Utilities Network Management System Quick Install Guide*
- *Oracle Utilities Network Management System Installation Guide*
- *Oracle Utilities Network Management System Licensing Information User Manual*
- *Oracle Utilities Network Management System User's Guide*
- *Oracle Utilities Network Management System Configuration Guide*
- *Oracle Utilities Network Management System Adapters Guide*
- *Oracle Utilities Network Management System Release Notes*

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.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

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 Pre-requisite Software.
3. Install the third-party software.
4. Unzip the Oracle Network Management System Operations Mobile Application project files from the $CES_HOME/sdk/nms_crew.zip file to your build environment system.
5. Install the Cordova Platforms and Plugins
6. Build the Oracle Network Management Systems Operations Mobile App for each of the desired platforms.
7. Run the compiled client using the browser to test your built application.
8. 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 maps and index
2. Setting you default server URI
3. Creating symbol files (.svg)
4. Mapping map objects to symbols (devices, conductors, conditions).
5. Configuring the conditions requests from the server
6. Configuring event_resources
7. Configuring da_resources
8. Configuring nms_resources.js

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 11.x or 12.x.
Android Tablets or Phones	Android device running Android 6.x, 7.x, or 8.x.
Windows PC or Tablet	Windows device running Window 10 (Build 1803+) 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.)	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.

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 (v8.11.1+) a platform built on Chrome's JavaScript runtime for building fast, scalable network applications.
- Git (v2.17.1+) a repository which offers all of the distributed revision control and source code management functionality in order to have the mobile standard plug-ins.
- Gradle (v4.6+) for automating the software build processes.
- Cordova CLI (v8.0.0). Cordova is a mobile development framework for enabling programmers to develop applications in HTML5, JavaScript and CSS3 instead of relying on platform-specific APIs like those in iOS or Android. Cordova enables wrapping up of HTML, CSS and JavaScript code depending upon the platform of the device.
- Oracle WebLogic 12g for the NMS Mobile Gateway
- Android SDK for creating new applications for the Android operating system. The recommended level is Android SDK Level 21.
- Xcode 8.3.3+ or greater and an Apple Developer ID for developing iOS applications using iOS SDKs.
- Microsoft Visual Studio 2015 or above.

The Apache Cordova website contains many resources and tutorials on the installation and usage of the Apache Cordova development and runtime processes. Please refer to the Apache Cordova website: http://cordova.apache.org. Go to the Documentation link and complete The Command-Line Interface Installation. To install a specific version of Cordova, add the version to the end of the install command; for example, to install version 8.0.0, use this command:

```
npm install -g cordova@8.0.0
```

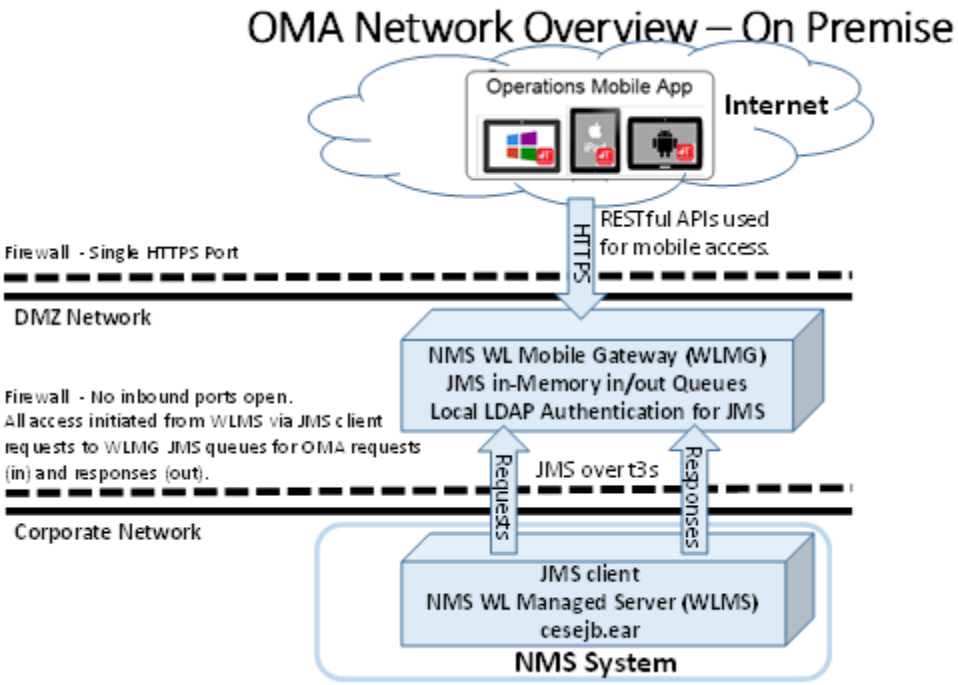
Chapter 3

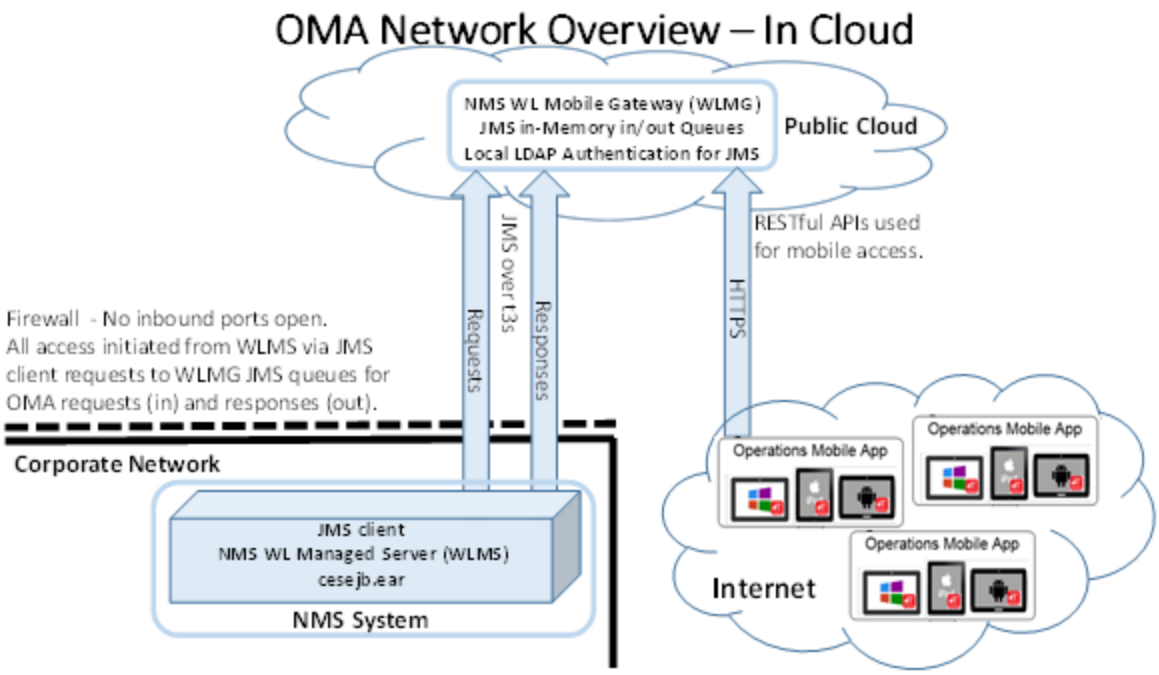
Mobile Gateway Server Installation

- **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:





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 $CES_HOME/dist/install directory and when the nms-install-config --java is run, the deployable Oracle Network Management System Mobile Gateway will reside in $NMS_HOME/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 30000
     - 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 changing the JNDI name to jms/oma-to-nms. Both the factory and the queue are deployed to a subdeployment to the JMSServer-oma.

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 server domain 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 server than the cesejb.ear, you will need to do the remaining steps in this section.

5. Do the following on the nms-ws Managed Server:

   - 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.
   - Setting for the managed server:
     - Configuration tab/General sub-tab Listening Port Enabled not checked and SSL Listening Port Enabled checked with a valid SSL Listen Port specified.
     - Configuration tab/General sub-tab Cluster should be set to Stand-Alone.
     - Protocols/General Enable Tunneling checked.
     - Protocols/Channels. 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 communications to external system and will require https. If this is not the case, choose t3s.

       - Click **Next**.
       - **Listen Address:** use the same address as the main listing 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. Do the following on the NMS Managed Server:
   - 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. It should be in the format `t3s://somehost:port`

     Notes:
     - Some corporate firewalls will not allow t3s communications to external system and will require https. If this is not the case, use https:// instead of t3s.
     - 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 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 the WebLogic configuration file named: config.xml

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**
- **Identity Cloud Service Provider**
- **Mobile Application Patches from the NMS Server**

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.

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 geojson files, a zipped version of all the geojson files, mapset.zip, will also be created in the $OPERATIONS_MODELS/export directory. This file will be used by the Operations Mobile Application if it chooses to use the option of 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.ces`**, as a template for your script to create GeoJSON files:

1. Copy `${CES_HOME}/OPAL/bin/`**`OPAL_build_mobile_maps.ces`** to `${CES_HOME}/<`*`project`*`>/bin/<`*`project`*`>_build_mobile_maps.ces`.

2. Edit the <*project*>_build_mobile_maps.ces 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 this script from your <*project*>_postbuild.ces script.

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.

Please use the template provided in the following location:

```
$CES_HOME/OPAL/sql/OPAL_geojson_export.dat
```

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 (-0PEN, -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.

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:

- The app can be built with nms_crew/www/data/geojson_maps.zip and nms_crew/www/data/mobile_geojson_maps.json files; when the app is first launched, these map files will be analyzed against any previously installed maps. The latest map versions will be used at runtime.

  To get the GeoJSON and index files into the app, in the build environment, zip the GeoJSON files into a geojson_maps.zip file and copy this zip file and the index file from the NMS server $OPERATIONS_MODELS/export directory to the nms_crew/www/data directory before building the app with Cordova.

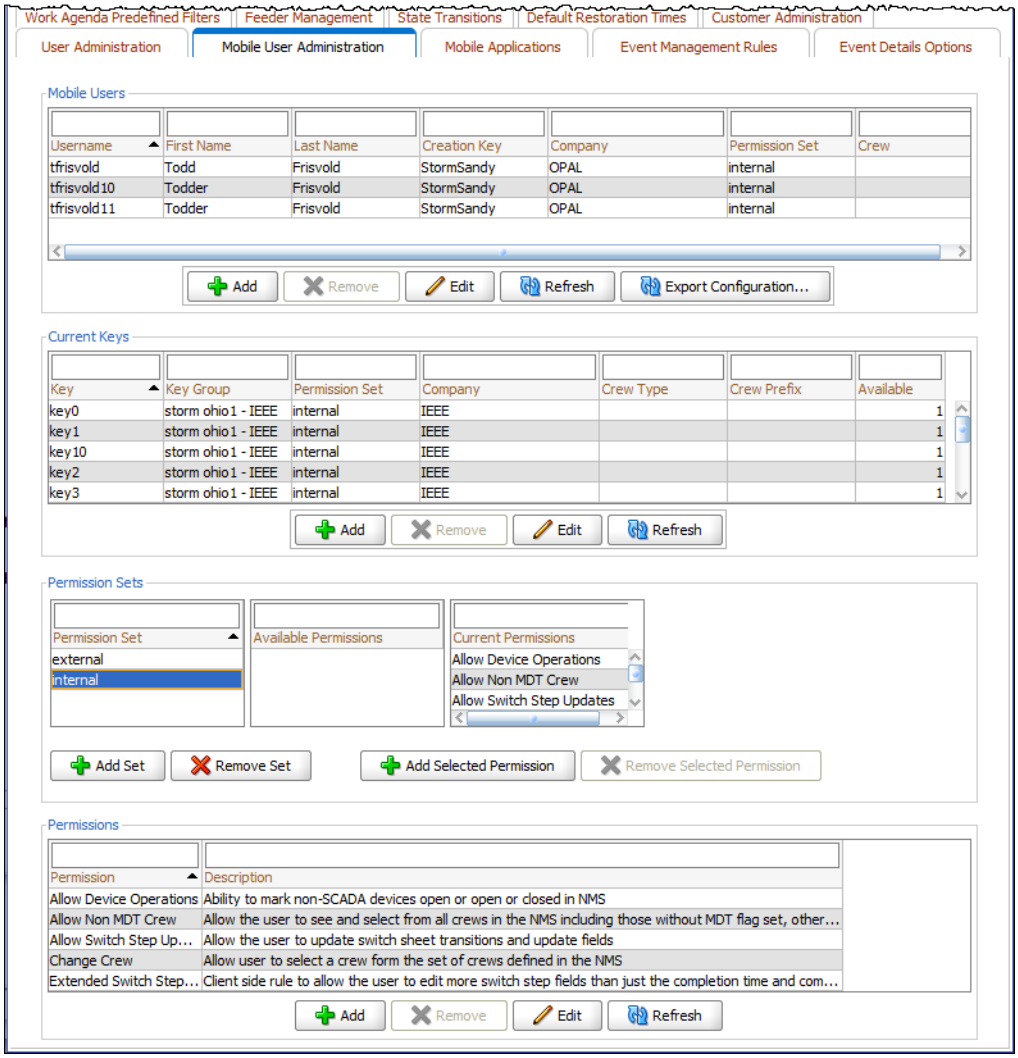
- At any time, you can go to the map librarian section of the OMA map page and request an updated server map index file (mobile_geojson_maps.json), and compare it to the already installed files, and download any outdated GeoJSON map files. OMA will also automatically check the server for updated maps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

  > **Note:** Do not include any <*mapname*>.geojson files in the application package nms_crew/www/data directory. This legacy practice has been obsoleted in favor of the geojson_maps.zip file process defined above.

For browser deployment of OMA, which is for testing only, the process to deploy the GeoJSON maps is different than above. Install the browser www directory on an http:// file server. Copy into the www/data directory the GeoJSON files and index file from the NMS server $OPERATIONS_MODELS/export 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.



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.
- **Allow Non MDT Crew:** This allows the user to select a Non MDT crew. If not enabled, then only MDT crews are available to the user.
- **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.

Two simple permissions sets are typical in configuration:

- Internal permission set with both Allow Non MDT Crew and Change Crew permissions
- External permission set with no permissions

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.

Create 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.

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.

Identity Cloud Service Provider

Overview

The Oracle Network Management System Operations Mobile App supports login using the Oracle Identity Cloud Service (IDCS). IDCS requires the same configuration as the LDAP/AD user validation in the CentricityServer.properties file described in the Using LDAP/AD User Validation section.

Identity Cloud Service – Setup

You will need to define two Applications in the Identity Cloud Service:

1. OMA Authentication Application

   Define an Application for the OMA client to authenticate against. Allowed Grant types should include Authorization Code and Implicit, Allow Non-HTTPS URLs checked, and Redirect URL to a existing URL (for example, https://google.com). This application will be used to configure the OMA Client in the nms_resources.js file:

   ```
   IDCS_HOST: "https://idcstrial09.identity.oraclecloud.com",

   IDCS_REDIRECT_URL: "https://google.com",

   IDCS_CLIENT_ID: "1a11a11a11111a11aa1a11a1a1111a1a",
   ```

2. NMS Authentication Application

   Define an Application for the NMS WLS to use to authenticate credentials via the IDCS Integration Provider. This Application will contain both a Client ID and a Client Secret. The Client Configuration will have the Register Client Checked, Allowed Grant Types will include (Resource Owner, Client Credentials, SAML2 Assertion, Refresh Token, Authorization Code, Implicit), Allow non-HTTPS URLs checked, redirector URL to an existing URL (for example, https://google.com), Client Type: Confidential, Trust Scope: Allowed scopes, Add a scope: NMSWLS/OMATest, Grant the Client Access to the Identity Cloud Service APIs (Cloud Gate, Application Administrator, Me, User Administrator). Resources will include Registered Resources selected, Access Token Expiration set to 604800, Primary Audience set to the OMA Application Name.

3. Define Users and Groups

   Users and Groups can be defined in IDCS directly or by configuring IDCS to connect to another authentication provider to get the user credentials and group membership. If using IDCS to define the users and groups, define the users and groups using the same group requirements as configured for the LDAP/AD in the CentricityServer.properties file.

NMS WebLogic Managed Server – IDCS Integration Provider

In the NMS WebLogic Managed Server, the Oracle Identity Cloud Integrator Authentication Provider will need to be added to your security realm:

1. Go to domain structures/Security Realms and select **myrealm**.
2. Select the Providers Tab
3. Click t the **New** button and give a name (for example, IDCS) and select **Type OracleIdentityCloudIntegrator**.
4. Set the **Control Flag** to **Optional** or **Sufficient** depending on whether additional providers are to be processed after this provider.
5. Set **Active Types** to include idcs_user_assertion, Idcs_user_assertion, and Authorization in the **Chosen** box, you can leave REMOTE_USER in the **Available** box.

6. Set the Provider Specific values as follows:

   Host: identity.oraclecloud.com (Base name of the IDCS server https://idcstrial09**.identity.oraclecloud.com**, not including the left most component, which is the hostname to be user later).

   Port: 443

   SSLEnabled: Checked

   Tenant: hostname (The host name from the IDCS Cloud Server: https://**idcstrial09**.identity.oraclecloud.com)

   Client Id: Hexadecimal string from the IDCS Application Configuration General Information Section

   Client Secret: Hexadecimal string copied from the IDCS Application Configuration Information Section

   Confirm Client Secret: Same as Client Secret

   Client Tenant: Leave Blank

   All other values can remain defaulted.

7. Restart the NMS WebLogic Managed Server

Configure the OMA Client

The OMA Application will need to know about the IDCS configuration. Please set the nms_resources.js file values to match the above IDCS configuration:

IDCS_HOST: "https://idcstrial09.identity.oraclecloud.com",

IDCS_REDIRECT_URL: "https://google.com",

IDCS_CLIENT_ID: "1a11a11a11111a11aa1a11a1a1111a1a",

Mobile Application Patches from the NMS Server

Overview

The Oracle Network Management System Operations Mobile App supports downloading updated applications from the NMS server without having to re-install on the Operations Mobile App device. Support for this functionality is available on iOS and Android devices only. Windows 10 support of this feature will be only provide version update notification but will not allow downloading of application patches to the device

Application Patch Generation

The Application Patch files are a result of the Cordova Build process. For each Architecture, the following command will build the Mobile App:

- **Browser:** `cordova build browser`
- **Android:** `cordova build android`
- **Windows 10:** `cordova build windows --archs=x64`
- **iOS:** `cordova build ios --device --release --codeSignIdentity="your ID" --developmentTeam=your-team-id --packageType=your-package-type --provisioningProfile=your-profile-guid`

The result of the build will produce the patch content.

- For browser, there is no download app support; however, a file with the patch information can be used to identify an out of date browser version.
- For Android, the nms_crew/platform/android/assets/www directory contains the patch contents.
- For Windows, the nms_crew/platform/windows/www directory contains the patch contents; however, there is no download app support for Windows, this patch content can be used to support the patch notification functionality.
- For iOS, the nms_crew/platform/ios/www directory contains the patch contents.

Zip the deployment contents into the Patch file Deployment name/location below.

Application Patch File Deployment

Zip up the patch contents into a file with this format:

`APP_{AppName}_{AppArchitecture}_{AppVersion}.zip`

Where:

- **AppName** is the app name, you can have many versions (for example: OMA, OMA-MA). The name must be alphanumeric and may include dashes (-), but not any other special characters.
- **AppArchitecture** is one of the following: ANDROID, IOS, WIN10, BROWSER
- **AppVersion** is a non-negative integer.

Examples

`APP_OMA_ANDROID_150.zip, APP_OMA-MA_WIN10_12.zip, APP_OMA_IOS_200.zip, APP_OMA_BROWSER_1.zip`

The Patch file will be placed in the $OPERATIONS_MODELS/apps directory.

Patching the Device

Refer to the NMS User Guide for instructions on patching the Operations Mobile Application from the device. If you want to force the application to update to the latest version, there is a line of code in the nms_crew.zip template in the index.js file that can be uncommented to force the auto-update process:

```
// Uncomment the following line if you want to force an automatic
update of the app if a newer version is available:
// if (window.localStorage["Settings:debugMode"] !== "true")
self.DownloadNewVersionClick();
```

Chapter 5

Client Development Setup on OSX

This chapter describes installing, building, and testing the Operations Mobile Application using a Macintosh running OSX.

- **Install Software**
- **Build Operations Mobile Application**
- **Testing**

Install Software

- **Install Prerequisite Software**
- **Install Operations Mobile App SDK**

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.

You may need to use proxy settings in order to get the third party software to work through your corporate network. Here are suggested environment variable to use, will need adjustments to match your corporate network addresses and ports.

```
http_proxy=http://www-proxy.us.oracle.com:80
HTTP_PROXY=http://www-proxy.us.oracle.com:80
https_proxy=http://www-proxy.us.oracle.com:80
HTTPS_PROXY=http://www-proxy.us.oracle.com:80
NPM_CONFIG_proxy=http://www-proxy.us.oracle.com:80
NPM_CONFIG_http_proxy=http://www-proxy.us.oracle.com:80
NPM_CONFIG_https_proxy=http://www-proxy.us.oracle.com:80
```

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 `$CES_HOME/sdk/nms_crew.zip` file of your Oracle Network Management System. Copy this directory to your development build environment system and unzip it. This will be your Cordova project directory.

Build Operations Mobile Application

Put gradle, node, and the android tools in your path:

```
export PATH=/Users/gradle/bin:$PATH:/usr/local/bin:/ Users/
appbuild/node/bin:$ANDROID_HOME/tools
```

Prior to building the target device application platforms, copy the geojson_maps.zip and mobile_geojson_maps.json files from the NMS server to the nms_crew/www/data directory and remove any *.geojson files from the same directory.

Prior to building the browser version, remove the geojson_maps.zip and mobile_geojson_maps.json files from the nms_crew/www/data directory.

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

Using a terminal window on the mac, change to your Cordova project directory (*i,e.*, /Users/appbuild/nms_crew):

```
cd /Users/appbuild/nms_crew
```

Look for the oma_build_ios.sh script.

You will need to edit this script set proxy settings if required. Proxy settings are at the top of the script, it will look something like this:

```
# This script is setup to run on OSX
# it is assumed NodeJs, Git, and cordova have all be installed.

# if needing a proxy to get to the internet, set the values here:
export _proxy_host=http://www-proxy.us.oracle.com
export _proxy_port=80

export
SEPARATOR="##########################################################
################"

echo $SEPARATOR
echo "Setting proxy env vars to: ${_proxy_host}:${_proxy_port}..."
export proxy=${_proxy_host}:${_proxy_port}
export PROXY=${_proxy_host}:${_proxy_port}
export http_proxy=${_proxy_host}:${_proxy_port}
export HTTP_PROXY=${_proxy_host}:${_proxy_port}
export https_proxy=${_proxy_host}:${_proxy_port}
export HTTPS_PROXY=${_proxy_host}:${_proxy_port}
export NPM_CONFIG_proxy=${_proxy_host}:${_proxy_port}
export NPM_CONFIG_http_proxy=${_proxy_host}:${_proxy_port}
export NPM_CONFIG_https_proxy=${_proxy_host}:${_proxy_port}
```

```
export GRADLE_OPTS="-Dhttp.proxyHost=${_proxy_host#*//} -
Dhttp.proxyPort=${_proxy_port} -Dhttps.proxyHost=${_proxy_host#*//} -
Dhttps.proxyPort=${_proxy_port}"


# big map builds:
export GRADLE_OPTS="-Dhttp.proxyHost=${_proxy_host#*//} -
Dhttp.proxyPort=${_proxy_port} -Dhttps.proxyHost=${_proxy_host#*//} -
Dhttps.proxyPort=${_proxy_port} -Dorg.gradle.jvmargs=\"-Xmx5000m\""
```

If you need to use a proxy to get to the Internet, change the first two export lines to match your proxy server requirements.

If you do no need to use a proxy, comment out all the proxy lines.

Then change the script to build the desired targeted platforms. By default, the script will try to build all three platforms:

```
echo "Building platforms..."
cordova build browser
echo $SEPARATOR
cordova build android
echo $SEPARATOR
# Replace the following line with your version that includes details about
# your developers license
cordova build ios --device --release --codeSignIdentity="iPhone Distribution: OPAL Corporation" --
developmentTeam=88XXX888XX --packageType=enterprise --provisioningProfile=x88x8x8x-xxxx-
8x88-x888-8x8xx8xxxx88
```

If you do not want to build a given platform, please comment out the corresponding "cordova build" line.

Then run the script to build the Operations Model Application.

Testing

- **Test with Safari Browser**
- **Test with iOS Device**
- **Test with Android Device**

Test with Safari Browser

To test with Safari on the Mac platform, you will need to start an https server in the project directory platforms/browser/www. To install and run the Node JS http server, use these commands:

```
$ cd platforms/browser/www

$ sudu npm install http-server -g

$ openssl req -newkey rsa:2048 -new -nodes -x509 -days 3650 -keyout key.pem out cert.pem

$ http-server -S -C cert.pem
```

This will start an https server in the local directory on all IPs associated with the system. The http-server command will report the IP addresses and posts it is serving from (ie https://127.0.0.1:8080).

OMA on a browser requires map files to be in the www/data directory. From another terminal, goto the platforms/browser/www/data directory and run the command:

```
$ unzip geojson_maps.zip
```

To run OMA, bring up Safari and g oto the address identified by the http-server:

```
https://127.0.0.1:8080
```

It will popup messages that the connection is not private. Go to the details and select the link to visit the website. Enter required information on subsequent prompts to allow access to the website.

The Operations Mobile Application main launch screen should appear.

Go to the settings screen:

- Set the Server/Mobile RESTFul URI.
- Set the Client/Server- periodicity for GPS update to a large value (ie 5000).

Go to a second tab in Safari and go to the address you provided as the Mobile RESTFul URI. The browser will inform you of security issues on going to that address. Proceed to the address by viewing the details and adding an exception. A login prompt will eventually appear, but you can just leave it without entering any information.

Go back to the tab running the Operations Mobile Application. Hit the Home button on the header bar to go to the launch page. Hit the Login button. You should now be able to run the Operations Mobile Application.

Follow the steps in the OPAL Operations Mobile Application Tests chapter.

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

Chapter 6

Client Development Setup on Windows

This chapter describes installing, building, and testing the Operations Mobile Application using a PC running Microsoft Windows.

- **Install Software**
- **Build Operations Mobile Application**
- **Testing**

Install Software

- **Install Prerequisite Software**
- **Install Operations Mobile App SDK**

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 Windows 10 or newer platform, install the Microsoft Visual Studio.

You may need to use proxy settings in order to get the third party software to work through your corporate network. Here are suggested environment variable to use, will need adjustments to match your corporate network addresses and ports.

```
http_proxy=http://www-proxy.us.oracle.com:80
HTTP_PROXY=http://www-proxy.us.oracle.com:80
https_proxy=http://www-proxy.us.oracle.com:80
HTTPS_PROXY=http://www-proxy.us.oracle.com:80
NPM_CONFIG_proxy=http://www-proxy.us.oracle.com:80
NPM_CONFIG_http_proxy=http://www-proxy.us.oracle.com:80
NPM_CONFIG_https_proxy=http://www-proxy.us.oracle.com:80
```

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 $CES_HOME/sdk/nms_crew.zip file of your Oracle Network Management System. Copy this directory to your development build environment system and unzip it. This will be your Cordova project directory.

Build Operations Mobile Application

Put gradle, node, and the android tools in your path using system setting or environment variables.

Prior to building the target device application platforms, copy the geojson_maps.zip and mobile_geojson_maps.json files from the NMS server to the nms_crew/www/data directory and remove any *.geojson files from the same directory.

Prior to building the browser, remove the geojson_maps.zip and mobile_geojson_maps.json files from the nms_crew/www/data directory.

Build the target application platforms. Refer to the oma_build.sh script in the nmc_crew.zip file. It is recommended to build OMA on Windows using the GIT bash shell, which is part of the required OMA prerequisite software. Copy your geojson_maps.zip file and mobil_geojson_maps.json file to the www/data directory,

Testing

- **Test with Chrome or Firefox Browser**
- **Test with Android Device**
- **Test with Windows 10**

Test with Chrome or Firefox Browser

Browsers have tight security and requires the OMA app to be started targeting OMA from a website. The easiest way to do this is to copy the cordova platforms/browser/www to a web https server, copy your geojson_maps.zip file and mobil_geojson_maps.json file to the www/data directory, and start it from there opening the www/index.html file.

You can use NodeJs to do this. To install the NodeJs http server, run this command:

```
npm install http-server -g
```

Go to the platforms/browser/www and create ssl certificates using this command:

openssl req -newkey rsa:2048 -new -nodes -x509 -days 3650 -keyout key.pem out cert.pem

and to start the server:

```
http-server -S -C cert.pem
```

This will start an https server in the local directory on all IPs associated with the system. The http-server command will report the IP addresses and posts it is serving from (ie https://127.0.0.1:8080).

OMA in a browser requires map files to be in the www/data directory. From another terminal, go to the platforms/browser/www/data directory and run the command:

$ unzip geojson_maps.zip

To run OMA, bring up your browser and goto the address identified by the http-server:

https://127.0.0.1:8080

You may also need to install the https: certificate for the NMS system you are attempting to connect to for the server.

This will open the application splash screen. Follow the steps in the OPAL Operations Mobile Application Tests chapter.

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.

Chapter 7

Client Development Setup on Linux

This chapter describes installing, building, and testing the Operations Mobile Application using a PC running Linux.

- **Install Software**
- **Build Operations Mobile Application**
- **Testing**

Install Software

- **Install Prerequisite Software**
- **Install Operations Mobile App SDK**

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. You may need to use proxy settings in order to get the third party software to work through your corporate network. Here are suggested environment variable to use, will need adjustments to match your corporate network addresses and ports.

```
http_proxy=http://www-proxy.us.oracle.com:80
HTTP_PROXY=http://www-proxy.us.oracle.com:80
https_proxy=http://www-proxy.us.oracle.com:80
HTTPS_PROXY=http://www-proxy.us.oracle.com:80
NPM_CONFIG_proxy=http://www-proxy.us.oracle.com:80
NPM_CONFIG_http_proxy=http://www-proxy.us.oracle.com:80
NPM_CONFIG_https_proxy=http://www-proxy.us.oracle.com:80
```

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 $CES_HOME/sdk/nms_crew.zip file of your Oracle Network Management System. Copy this directory to your development build environment system and unzip it. This will be your Cordova project directory.

Build Operations Mobile Application

Put gradle, node, and the android tools in your path using system setting or environment variables.

Prior to building the target device application platforms, copy the geojson_maps.zip and mobile_geojson_maps.json files from the NMS server to the nms_crew/www/data directory and remove any *.geojson files from the same directory.

Prior to building the browser, remove the geojson_maps.zip and mobile_geojson_maps.json files from the nms_crew/www/data directory.

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

Change to your Cordova project directory (*i,e.*, /Users/appbuild/nms_crew):

cd /Users/appbuild/nms_crew

Look for the oma_build_ios.sh script.

You will need to edit this script set proxy settings if required. Proxy settings are at the top of the script, it will look something like this:

```
# This script is setup to run on OSX

# it is assumed NodeJs, Git, and cordova have all be installed.

# if needing a proxy to get to the internet, set the values here:

export _proxy_host=http://www-proxy.us.oracle.com

export _proxy_port=80

export
SEPARATOR="##############################################################
#######################"

echo $SEPARATOR

echo "Setting proxy env vars to: ${_proxy_host}:${_proxy_port}..."

export proxy=${_proxy_host}:${_proxy_port}

export PROXY=${_proxy_host}:${_proxy_port}

export http_proxy=${_proxy_host}:${_proxy_port}

export HTTP_PROXY=${_proxy_host}:${_proxy_port}

export https_proxy=${_proxy_host}:${_proxy_port}

export HTTPS_PROXY=${_proxy_host}:${_proxy_port}

export NPM_CONFIG_proxy=${_proxy_host}:${_proxy_port}

export NPM_CONFIG_http_proxy=${_proxy_host}:${_proxy_port}

export NPM_CONFIG_https_proxy=${_proxy_host}:${_proxy_port}

export GRADLE_OPTS="-Dhttp.proxyHost=${_proxy_host#*//} -
Dhttp.proxyPort=${_proxy_port} -Dhttps.proxyHost=${_proxy_host#*//} -
Dhttps.proxyPort=${_proxy_port}"

# big map builds:

export GRADLE_OPTS="-Dhttp.proxyHost=${_proxy_host#*//} -
Dhttp.proxyPort=${_proxy_port} -Dhttps.proxyHost=${_proxy_host#*//} -
Dhttps.proxyPort=${_proxy_port} -Dorg.gradle.jvmargs=\"-Xmx5000m\""
```

If you need to use a proxy to get to the Internet, change the first two export lines to match your proxy server requirements.

If you do no need to use a proxy, comment out all the proxy lines.

Then change the script to build the desired targeted platforms. By default, the script will try to build all three platforms:

```
echo "Building platforms..."
cordova build browser
echo $SEPARATOR
cordova build android
echo $SEPARATOR
# Replace the following line with your version that includes details about
# your developers license
cordova build ios --device --release --codeSignIdentity="iPhone Distribution: OPAL Corporation" --developmentTeam=88XXX888XX --packageType=enterprise --provisioningProfile=x88x8x8x-xxxx-8x88-x888-8x8xx8xxxx88
```

We only support building browsers on Android or Linux, so comment out the "cordova build ios" line. If you do not want to build any other platform, please comment out the corresponding "cordova build" line.

Then run the script to build the Operations Model Application.

Testing

- **Test with Android Device**

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

Chapter 8

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 Appls 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 Window 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 9

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 MBService startup parameters to include the –export option. This can be done in the $NMS_HOME/etc/system.dat file:

```
program  MBService       MBService       -dbname mb –export
```

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

2. Restart MBService with net parameter:

```
$ Action any.MBService stop
$ sms_start.ces -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.ces
```

4. Setup an Operations Mobile Application development environment on OSX, Windows 10, or Linux as describe in this document.

5. Copy the OPAL geojson maps and index file to your Operations Mobile Application development environment nms_crew/www/data directory. The files can be found on the NMS server in this location:

```
$OPERATIONS_MODELS/export/*json
```

6. Configure the default NMS Mobile Gateway URL into the development environment file nms_crew/www/js/settingsDefaults.js file by changing this line using your server and port:

```
self.serverMobileRestURI = ko.observable('https://localhost:7082/nms-
ws', {persist: 'Settings:serverMobileRestURI_string_'});
```

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 10

Operations Mobile Application Project Setup

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

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

   ```
   program  MBService       MBService       -dbname mb -export
   ```

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

2. Restart MBService with net parameter:

   ```
   $ Action any.MBService stop
   $ sms_start.ces -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' and coord_system = 0;"`
   do
     DiagramBuilder -map ${f%.mad}
   done
   $ <project>_build_mobile_maps.ces
   ```

5. Setup an Operations Mobile Application development environment on OSX, Windows 10, or Linux as describe in this document.

6. Copy the geojson maps and index file to your Operations Mobile Application development environment nms_crew/www/data directory. The files can be found on the NMS server in this location:

   ```
   $OPERATIONS_MODELS/export/*json
   ```

7. In the development environment, configure the default NMS Mobile Gateway URL into the development environment file nms_crew/www/js/settingsDefaults.js file by changing this line using your server and port:

   ```
   self.serverMobileRestURI = ko.observable('https://localhost:7082/
   nms-ws', {persist: 'Settings:serverMobileRestURI_string_'});
   ```

8. In the development environment, configure your device symbology. Put the device .svg files in the nms_crew/www/img directory. Map the files to the geojson object SYMBOL attribute in the nms_crew/www/js/mapo.js file using the AddMarkerToElectricBucketStyle function.

9. In the development environment, configure your conductor symbology. Map the symbology to the geojson object SYMBOL attribute in the nms_crew/www/js/mapo.js file using the AddLineToElectricBucketStyle function.

10. In the development environment, configure your condition symbology. Put the device .svg files in the nms_crew/www/img directory. Map the files to the condition class number and

condition status number attributes in the nms_crew/www/js/mapo.js file using the AddMarkerToConditionBucketStyle function.

11. In the development environment, configure the classes of interest to get from the server. In the nms_crew/www/js/mapo.js file, locate the ajax call and adjust the list of condition types you want to use. The line will look like this:

```
// make ajax call to get events
var ajaxURL =
window.localStorage["Settings:serverMobileRestURI_string_"] +
"/mobile/conditions?qt=event,da,incident,"+
"assessment,note,tag,info,clear,hold,hot,disable,wire_down,"+
"warn,ground,dcz,associated_document&long1=" +
      mapMBR.getMaxX() + "&lat1=" + mapMBR.getMaxY() +
 "&long2=" + mapMBR.getMinX() + "&lat2=" + mapMBR.getMinY();
```

12. In the development environment, configure the event_type_labels in the nms_crew/www/js/event_resources.js file to match your project:

```
 // labels for NMS event types
  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 nms_crew/www/js/event_resources.js file to match your project:

```
// labels for NMS critical customer types
  customer_type_labels: {
    0: 'Standard',
    1: 'Emergency',
    2: 'Key',
...
```

14. In the development environment, configure the picklist_type in the nms_crew/www/js/event_resources.js file to match your project:

```
// Event Details
  // GUI types
  // matches PICK_ENV_MAPPING in JBotFormat_en_US.properties
  picklist_type: {
    NO_OUTAGE: 'radial',
    PROBABLE_SERVICE_OUTAGE: 'radial',
    PROBABLE_DEVICE_OUTAGE: 'radial',
...
```

15. In the development environment, configure the picklist_filters in the nms_crew/www/js/event_resources.js file to match your project:

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

16. In the development environment, configure the picklist_cfg in the nms_crew/www/js/event_resources.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)
  picklist_cfg: {
    system_om:        {label:'System',visible:true,required:false},
...
```

17. In the development environment, configure the default_crew_types in the nms_crew/www/js/nms_resources.js file to match your project:

```
 // default crew types
  default_crew_types: [
    {name: 'Trouble', id: 1},
    {name: 'Service', id: 2},
    {name: 'Eval', id: 3},
    {name: 'Line', id: 4},
    {name: 'Tree Crew', id: 5}
  ]
```

18. In the development environment, configure the app_key, project_version, product_verison, download_version, download_application_name in the nms_crew/www/js/nms_resources.js file to match your project, keeping the formatting the same as the template:

```
// The key of this application (from Configuraiton Assistant)
  app_key: '31f7f96c-5f58-4070-b25b-8722a0b05f9d',

  // The project application version
    project_version: 'A - 06SEP2017',

  // The product application version
    product_version: '2.3.0.1.0',

 // download_version and download_application_name will be used to
create the
  // download .zip file in the format:
  //
APP_<APP_<download_application_name>_<architecture>_<download_version>
.zip
  // <architecture> can be one of the following: ANDROID, IOS,
WIN10.
  // BRWOSER is not supported for downloaded versions
  // ie APP_OMA_ANDROID_120.zip
  // The doaloaded application version - must be integer
    download_version: '250',
  // The doaloaded application version
    download_application_name: 'OMA',
```

19. In the development environment, configure the section_options in the nms_crew/www/js/da_resources.js file to match your project:

```
section_options: [
    'Service',
    'Secondary',
    'Lateral',
    'Backbone',
  ],
```

20. In the development environment, configure the location_options in the nms_crew/www/js/da_resources.js file to match your project:

```
location_options: [
    'Street',
    'Rear Lot Line',
    'Other',
  ],
```

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

Appendix A

Restricted Use and User License Terms

- **Mobile Archive Restricted Use**
- **Mobile Application End 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.

Mobile Application End User License Terms

Any Mobile Application distribution to End Users must be subject to a legally binding end user license agreement (the "EULA") between you and each End User pertaining to the Mobile Application that must, at a minimum, contain the following terms:

(a) Include acknowledgments by you and the End User that the EULA is concluded between you and the End User only and that the following apply:

(i) you are solely responsible for each Mobile Application's content, maintenance, and support; and

(ii) you are solely responsible for addressing, settling, and discharging any claims of the End User or any third party relating to the Mobile Application or the End User's possession and/or use of that Mobile Application, including, but not limited to product liability claims; any claim that the Mobile Application fails to conform to any applicable legal or regulatory requirement; any claims arising under consumer protection or similar legislation; and any claims that the End User's possession and use of that Mobile Application infringes a third party's intellectual property rights;

(b) Provide only a non-transferable, terminable license to the End User that prohibits (i) modifying or creating derivative works or (ii) decrypting, decompiling, reverse engineering, disassembling or attempting to derive the Mobile Application source code (unless such actions are expressly permitted by applicable law);

(c) Notify the End User that the Mobile Application is subject to a restricted license and can be used only in conjunction with the specific Oracle-based solution(s) for which it is designed;

(d) Provide no limitation of your liability to the End User beyond what is permitted by applicable law;

(e) Require the End User to comply fully with all relevant export laws and regulations of the U.S. and other applicable export and import laws to assure that the Mobile Application, nor any direct products thereof, is exported, directly or indirectly, in violation of applicable laws;

(f) State in the EULA your name and address to which any End User questions, complaints or claims with respect to the Mobile Application can be directed;

(g) State in the EULA that the End User must comply with applicable third-party terms when using the Mobile Application and that third-party components that may be appropriate or necessary for use with the Mobile Application are specified in the documentation for that program (or as otherwise notified by you) and that those third party components are licensed to the End User only for use with the Mobile Application under the terms of the third party license agreement specified in the documentation for that program (or as otherwise notified by you) and not under the terms of the EULA;

(h) State that the licenses provided in the EULA automatically terminate upon breach of the EULA terms and in addition that the licenses provided in the EULA may be terminated upon notice;

(i) State that upon termination of the EULA the End User must discontinue all use of the Mobile Application and to delete all copies of the Mobile Application;

(j) Disclaim in the EULA, to the extent permitted by applicable law, a third party's liability for (a) any damages, whether direct, indirect, incidental, special, punitive or consequential, and (b) any loss of profits, revenue, data or data use, arising from use of the Mobile Application;

(k) Designate Oracle as a third party beneficiary. Oracle will have the right to enforce the EULA against the End Users; and

(l) State that your licensors retain all ownership and intellectual property rights in the Mobile Application.

You agree to inform Oracle promptly if you are aware of any breach of the EULA. You agree to be financially responsible to Oracle for all damages or losses caused by your failure to include the required contractual terms set forth above in each EULA between you and an End User.