

Oracle Utilities Network Management System

Operations Mobile Application Installation and Deployment Guide.

Release 2.5.0.2.5

F54379-05

June 2023

Oracle Utilities Network Management System Operations Mobile Application Installation and Deployment
Guide, Release 2.5.0.2.5

Content updated for Release 2.5.0.2.5

F54379-05

Copyright © 1991, 2023 Oracle and/or its affiliates.

Preface	i-v
Audience	i-v
Related Documents	i-v
Conventions.....	i-vi
Chapter 1	
Installation and Deployment Overview	1-1
Server Installation Overview	1-2
Client Development Installation Overview.....	1-2
Client Configuration Overview	1-3
Client Deployment Overview	1-3
Chapter 2	
Supported Platforms & Hardware Requirements	2-1
Hardware Requirements	2-1
Client Hardware Requirements.....	2-1
Server Hardware Requirements	2-1
Development Hardware Requirements	2-1
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	3-8
Configuring the Default Control Zone and Crew Defaults.....	3-8
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.....	4-5
Permissions and Permission Sets	4-6
Predefined Users	4-6
Creating Users Using a Key.....	4-6
Using LDAP/AD User Validation.....	4-7
Mobile Applications Validation	4-8
Server Based Documents	4-9
Overview	4-9
Document Locations	4-9
Document Configuration.....	4-9
Configuring OMA Object Attribute Viewer	4-10
Overview	4-10
Configuration	4-11
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
Identity Cloud Service Provider	4-17
Overview	4-17

Identity Cloud Service – Setup.....	4-17
URL with IDCS Login Script.....	4-18
NMS WebLogic Managed Server – IDCS Integration Provider	4-18
Configure the OMA Client.....	4-19
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.....	5-3
Client PWA Installation Instructions	5-5
Client PWA Update Instructions	5-7
Client PWA Uninstall Instructions	5-8
Windows - Edge.....	5-8
Removing a Chrome OMA PWA on Windows and Linux.....	5-9
Android.....	5-10
iOS.....	5-10
OMA Client Configuration.....	5-11
Migrating Map Configuration from Previous OMA Versions.....	5-11
Third Party Application Integration with the OMA Client.....	5-11
Chapter 6	
Client Deployment.....	6-1
Server Application Install Instructions.....	6-1
Server Application Update Instructions.....	6-2
Chapter 7	
OMA Native PWA Wrappers	7-1
Building and Deploying the OMA PWA Wrapper for Android	7-2
Building and Deploying the OMA PWA Wrapper for iOS	7-3
How to Issue a Local Notification from OMA When Using the PWA Wrapper.....	7-4
Chapter 8	
Operations Mobile Application Setup on OPAL.....	8-1
Chapter 9	
Operations Mobile Application Project Setup.....	9-1
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 Release 2.6.0.0.0 documentation set:

- *Network Management System Adapters Guide*
- *Network Management System Advanced Distribution Management System Implementation Guide*
- *Network Management System Configuration Guide*
- *Network Management System Installation Guide*
- *Network Management System Operations Mobile Application Installation and Deployment Guide*
- *Network Management System Licensing Information User Manual*
- *Network Management System OMS for Water User's Guide*
- *Network Management System Quick Install Guide*
- *Network Management System Release Notes*
- *Network Management System Security Guide*

-
- *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. Use the table in the [Hardware Requirements](#) section to identify the supported build environment platforms.
2. Review and prepare for the download and installation of required Oracle and third-party software as described in the [Prerequisite Software](#) section.
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. Install the Node.JS from <https://nodejs.org>. You will need version 14.17.3 or higher.
6. Build the Oracle Network Management Systems Operations Mobile App.
7. Run the built client using the browser to test your built application.

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 URIs 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

Provide website links to make OMA available to your users.

Chapter 2

Supported Platforms & Hardware Requirements

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

Hardware Requirements

Client Hardware Requirements

The Operations Mobile Application is supported on iOS tablets/phones, Android tablets/phones, Windows PC/tablets and Linux. Please see the "Required Unbundled Oracle and 3rd Party Products" in the *Oracle Utilities Network Management System License Information Manual* for supported OS/browser versions on each platform.

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 ceselj.ear. The nms-ws application server requires a minimum of 2 CPU cores and 8 GB of memory.

Development Hardware Requirements

You can develop OMA on most platforms supported by Node.JS and bash scripting. Recommended development platforms include Window 10, Oracle Linux, and MacOS.:

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 (v14.17.3+), 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

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>.

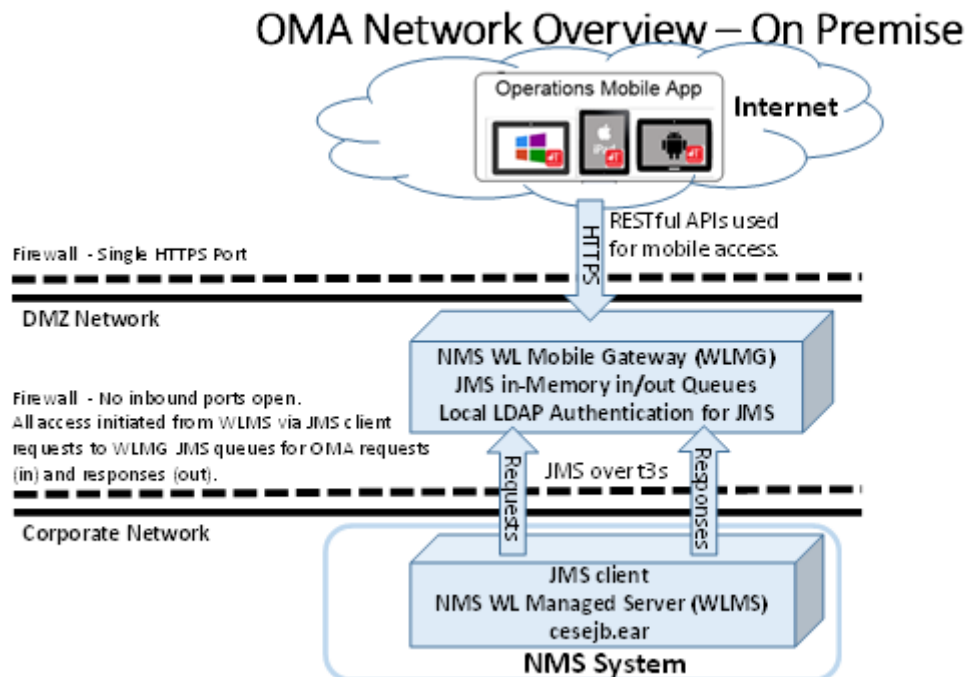
Chapter 3

Mobile Gateway Server Installation

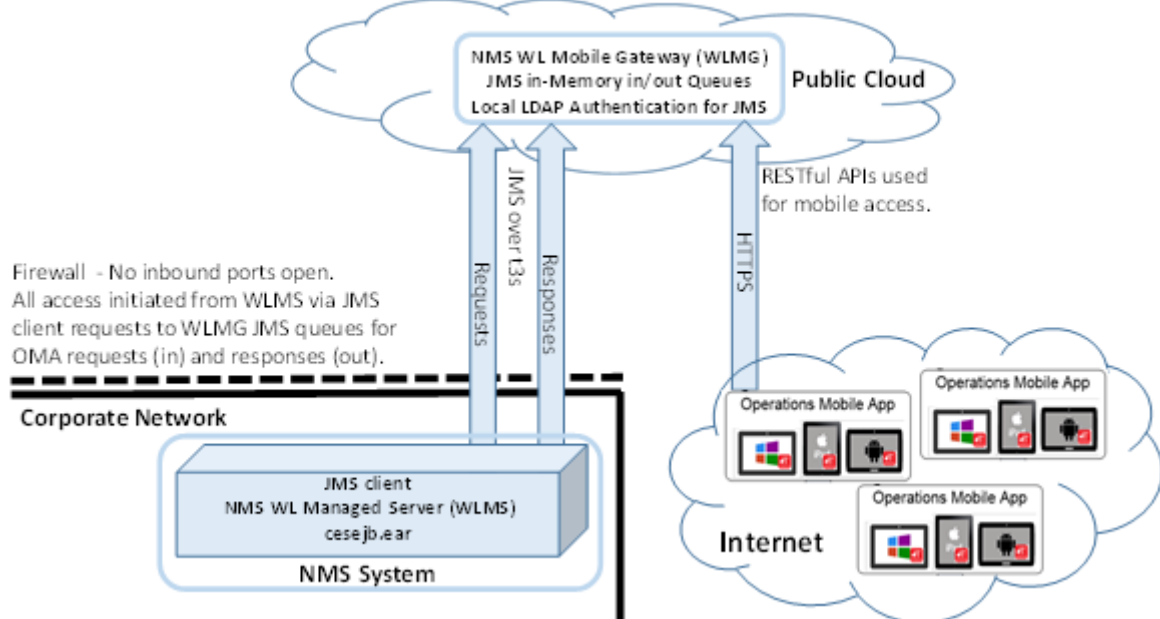
- [Deploy the Mobile Gateway](#)
- [Configuring WebLogic to Handle HTTP Basic Challenges Correctly](#)

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`.

There are many options for how the Oracle Network Management System Mobile Gateway is deployed to a WebLogic system and how they interact with the NMS WebLogic Managed Servers. Please refer to the "NMS Configuration Guidelines for Multiple WebLogic Managed Servers" section in the User Authentication chapter of the *Oracle Utilities Network Management System Configuration Guide*.

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. Locate the script `$NMS_BASE/dist/install/wls/scripts/oma-jms.py`. Transfer this to your WebLogic installation. Then locate the `wlst.sh` command so that you can provide the full path to it. Typically, this is in the `$WL_HOME/common/bin` directory. Then on the server that is running `nms-ws.ear`, run:

```
<path to>/wlst.sh oma-jms.py
```

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

When deploying to a cluster, deploy to a server within the cluster (not the cluster itself). 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 JMSServer-oma.

If this WebLogic instance is only to provide support for OMA and/or Flex, then disable measurement updating on that server. To do this, add the following WebLogic startup parameters:

```
-DscadaMeasureAnalogUpdates=0  
-DscadaMeasureDigitalUpdates=0
```

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 supported by the firewall, we recommend t3s.

- 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. To provide fault tolerance when connecting to a cluster or multiple servers, set up multiple oma gateways with each gateway pointing to a specific cesejb instance. Then have a load balancer in front of the oma gateways configured to fail over if a server goes unresponsive. The connection should be sticky so that if a client connects to a server, it should continue to use that server unless it becomes unavailable. To determine the health of an oma server, the load balancer can use this suggested call to this URL:

To verify an OMA server is up, periodically call this url:

```
https://host:port/nms-ws/mobile/version
```

That will return a json object similar to this:


```

{
  "serverInfo": {
    "codebase": "APPLICATION_SERVER",
    "properties": {
      "APPLICATION_SERVER_STATUS": "up",
      "APPLICATION_SERVER_CVS_TAG": "REL_2_5_0_2",
      "APPLICATION_SERVER_PROJECT_NAME": "OPAL",
      "APPLICATION_SERVER_PROJECT_BUILD_DATE": "2022:09:26
23:40:20",
      "APPLICATION_SERVER_PROJECT_TAG": "NO_TAG",
      "APPLICATION_SERVER_BUILD_DATE": "2022:09:26
23:13:45",
      "APPLICATION_SERVER_HOST_NAME": "nms-wls-host"
    }
  },
  "mobileInfo": {
    "codebase": "MOBILE_SERVER",
    "properties": {
      "MOBILE_SERVER_PROJECT_NAME": "OPAL",
      "MOBILE_SERVER_BUILD_DATE": "2022:09:26 23:13:45",
      "MOBILE_SERVER_PROJECT_BUILD_DATE": "2022:09:26
23:40:20",
      "MOBILE_SERVER_HOST_NAME": "nms-wls-host",
      "MOBILE_SERVER_PROJECT_TAG": "NO_TAG",
      "MOBILE_SERVER_CVS_TAG": "REL_2_5_0_2"
    }
  }
}

```

Look at the **APPLICATION_SERVER_STATUS**. It should be **up**; if it is down or no answer, then it should fail over.

7. 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 125. (This may need to be adjusted but as a starting point use 8 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. This should be configured for both the domain running cesejb.ear as well as nms-ws.ear (in the case when there are separate domains).
8. Do the following on the NMS Managed Server:
 - Verify the managed servers running cesejb.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 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`). These files are zipped into `offline_landbase_maps.zip`. In addition to this zip file, an `offline_landbase_maps.json` file is also created containing version information. These two files will need to be placed in your OMA2 project directory and they will be consumed in the OMA2 build process. 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 OMA2 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 main menu includes options like Event Details Options, Feeder Management, State Transitions, Default Restoration Times, Alarm Rules, and Customer Administration. The 'Mobile User Administration' tab is selected, showing a table of mobile users and options to add, remove, edit, refresh, export configuration, and view login history.

Mobile Users Table:

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		

Mobile User Type Memberships:

REMAINING MOBILE USER TYPES	CURRENT MOBILE USER TYPES
Damage Assessor	Full Operations
Hazard Responder	
View Only	

Current Keys Table:

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 - IEE	internal	Full Operations, View ...	IEEE		OMA_	1
key1	storm ohio1 - IEE	internal	Full Operations, View ...	IEEE		OMA_	1
key10	storm ohio1 - IEE	internal	Full Operations, View ...	IEEE		OMA_	1
key5	storm ohio1 - IEE	internal	Full Operations, View ...	IEEE		OMA_	1

Permission Sets:

PERMISSION SET	AVAILABLE PERMISSIONS	CURRENT PERMISSIONS
external	Access Events	Allow Device Operations
internal	Access Miscellaneous Log	Allow Switch Step Self Instruct
service	Allow as-user Parameter	Allow Switch Step Updates
	Confirm Outage	Change Crew
	Edit Permanent Crew	Send HLM
	Group Event	

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.

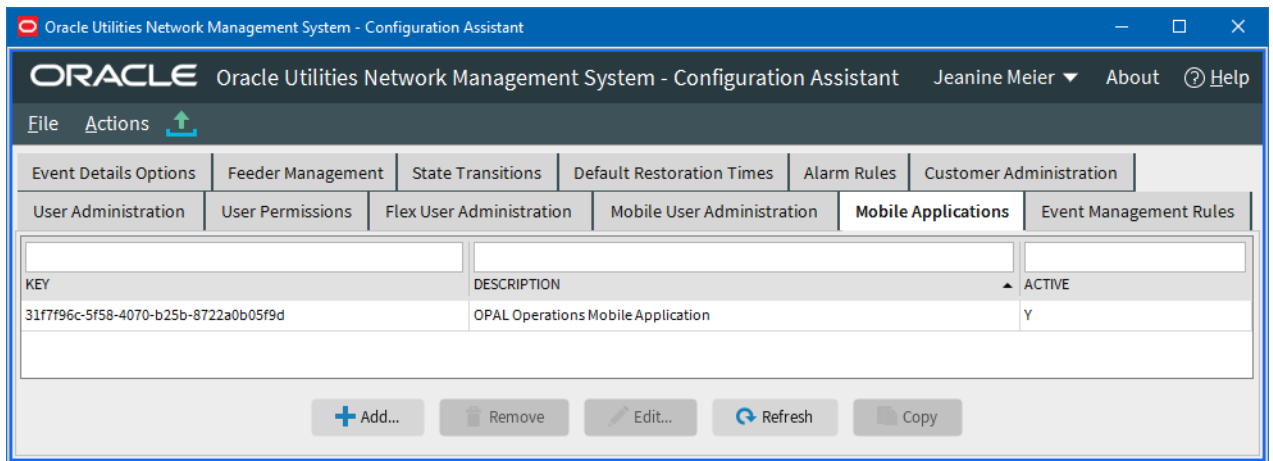
To configure LDAP/AD user validation for OMA, there are two lines that need to be enabled in the `CentricityServer.properties` file. Uncomment the last two lines, as shown below:

```
# 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).

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 the NMS server in the `$(OPERATIONS_MODELS)/docs` directory, add a `docs.json` file that described the document files to be presented to the OMA user. Here is an example of the file format:

```
{
  "documents": [
    {
      "name": "OPAL Safety Rules in Word Format",
      "description": "The Oracle Power and Light safety rules
document in word format.\nPlease read and follow these rules every
day!",
      "filename": "OPAL_safety_rules.doc",
      "version": "Rev 6"
    },
    {
      "name": "OPAL Safety Rules in PFD Format",
      "description": "The Oracle Power and Light safety rules
document in word format.\nPlease read and follow these rules every
day!",
      "filename": "OPAL_safety_rules.pdf",
      "version": "Rev 2"
    }
  ]
}
```

This will look like this in OMA:



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 an 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]_geojson_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]_geojson_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]_geojson_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 `$(OPERATIONS_MODELS)/export` directory to be processed by the `geojson` generation process.
- Change your model build post-process script (`[project]-postbuild`) to do schematic map generation before the OMA `geojson` map generation.
- Change your `[project]-build-mobile-maps` script to process `schematica .mb` files in the `$(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 users 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_num
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_RO;
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,
    offlineDataFile: 'offlineDeviceIdSearch.json'
  },
  ...
]
```

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 from the offlineSearchFiles.zip file. 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.

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

Redirect URL to a URL that runs a special IDCS login script required by OMA to get a message that the login was successful and return the login token back to the OMA application (for example, <http://frisvold.us/oracle/messagecode>).

This application will be used to configure the OMA Client in the `src\js\resources\config\loginSettings.js` file:

```
self.IDCS_HOST = 'https://idcstrial09.identity.oraclecloud.com';  
self.IDCS_REDIRECT_URL = 'http://frisvold.us/oracle/messagecode';  
self.IDCS_CLIENT_ID = '1a11a11a11111a11a11a11a1111a1a';
```

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

Authorized Resources: Specific

Resources/Add 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.

URL with IDCS Login Script

For OMA to get the IDCS login status and authorization token, you must provide a URL that the IDCS login process will call with the login status and token. This URL will contain a simple script that will analyze the parameters, pull out the login status and token, and message it back to OMA. An example of this can be found at:

`http://frisvold.us/oracle/messagecode`

and contains a one-page HTML document that looks like this:

```
<html>
  <head>
    <title>Untitled</title>
  </head>
  <body> <h2>postMessage of Code</h2>
  <script>console.log(window.location.search.substring(1));
  if (window.opener)
  window.opener.postMessage(window.location.search.substring(1),
  "");
  else window.postMessage(window.location.search.substring(1), "");
  </script> </body>
</html>
```

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 `src\js\resources\config\loginSettings.js` file values to match the above IDCS configuration:

```
self.IDCS_HOST = 'https://idcstrial09.identity.oraclecloud.com';
self.IDCS_REDIRECT_URL = 'http://frisvold.us/oracle/messagecode';
self.IDCS_CLIENT_ID = '1a11a11a11111a11a11a11a1111a1a';
```

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](#)
- [Client PWA Installation Instructions](#)
- [Client PWA Uninstall Instructions](#)
- [Client PWA Update Instructions](#)
- [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.

Install Operations Mobile App SDK

The Operations Mobile App SDK is located in the `$NMS_DIST/sdk/OMA2.zip` file of your Oracle Network Management System. Copy the zip file to your development build environment system and unzip it. 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] [options]
```

Also you may add the following options to set build mode:

- clean - clean out all build artifacts and exit without building
- noclean - do not clean out all build artifacts before starting the build
 - The DEFAULT is to clean out all build artifacts before building
- oma - build OMA, which is the default when building from sdk/OMA2.zip
- pwa - build as a PWA
- optimized - build optimized
- all - will build all permutations including:
 - oma web non-optimized
 - oma pwa non-optimized
 - oma web optimized
 - oma pwa optimized
- lock_packages <pkg-file>
 - Specify this to make the build use the versions specified in the package-lock.json file instead of the latest versions determined by the package.json file. Useful to ensure no third party dependencies change once development is complete and you want to ensure stability and reproducibility of the build

Also you may add the following options to alter proxy/registry information:

- proxy_url <url> - the URL for your proxy server (if required)
- proxy_port <port> - the port number for your proxy server (if required)
- registry_url <url> - The URL for an internal npm registry (if required)

Build applications are placed in the `./dist` directory in sub-directories

based on application type and options. For example:

```
dist/oma/www
```

```
dist/oma_pwa/www
dist/oma_optimized/www
dist/oma_pwa_optimized/www
```

Recommended use of each application type:

1. dist/oma/www - Best for Development
2. dist/oma_pwa/www - Best for Production
3. dist/oma_optimized/www and dist/oma_pwa_optimized/www are experimental and will be productized in a future release.

Testing

To build OMA using the non-PWA version (dist/oma/www) and to serve it from an http:// server:

1. Build OMA as a non-PWA non-Optimized app using this command:

```
./oma2_build.sh -oma
```

2. To serve OMA from a test http server, run this command:

```
npm run serve
```

OMA will open in a browser on your device at this address: `http://localhost:8000`

OMA running as a PWA requires it to be served from an HTTPS server. This is a general requirement of all PWA applications.

The HTTPS server must meet these requirements:

- It must be running on a system with a Fully Qualified Domain Name (FQDN; such as nms-host.oracle.com).
- The FQDN must be registered with your systems DNS server.
- The FQDN must appear in the Subject Alternate Name (SAN) attribute of the HTTPS SSL certificate.

The following is a summary of how to build OMA using the PWA non-optimized version and to serve it from an https:// server:

1. Build OMA as a non-PWA non-Optimized app using this command:

```
./oma2_build.sh -oma -pwa
```

Generate an https/ssl self signed certificate using your systems DNS registered FQDN. Here is an example of how to do that:

```
openssl req -new -newkey rsa:2048 -new -nodes -x509 -days 3650 -
passout pass:1234 -keyout key.pem -out cert.pem -subj "/C=US/
CN=nms-host.oracle.com" -addext "subjectAltName =
DNS:nms-host.oracle.com"
```

If you are running a very old version of openssl, it may not support the `-addext` option, you may need to update your system to a newer version of openssl. openssl added support for `-addext` in version 1.1.1 in 2018.

2. To serve OMA from a test http serve, run this command:

```
ojet serve web
```

OMA will open in a browser on your device at this address: `https://localhost:8000`. Change the address in the browser address bar to your FQDN (`https://nmshost.oracle.com:8000`).

3. In most cases, your browser will indicate in the address bar that the https/ssl certificate is not trusted. Install your SSL certificate into your systems trusted certificate store.

For Windows Systems

1. From the browser tab running OMA from the FQDA https server, right click in the address bar on the "Not Secure" area. From Chrome, select "Certificate is not valid" and the certificate details will be displayed. From Edge, click on "Your connection to this site isn't secure" and it will open a dialog box, click the certificate icon on the header bar, and the certificate details will open.
2. From the certificate details popup, goto the details tab and click **Copy to File...** and the Certificate Export Wizard will open. Click **Next**, select **DER encoded binary**, click **Next**, enter file path or browse to a file location and enter a name (C:\cert.cer), click **Next**, and then **Finish**.
3. From the Windows start menu, enter Certificates and click on Manage Computer Certificates. The certlm app will popup.
4. Select "Trusted Root Certificates" on the left column and from the Action menu, select All Tasks and then Import... The Certificate Import Wizard will appear.
5. From the Certificate Import Wizard dialog box, select **Next**, enter or browse to the file where you saved the certificate (for example, C:\cert.cer) and click **Next**. Select Place all certificates in the following store, then click **Browse...** and click "Trusted Root Certificate Authorities" and **OK**, then **Next**, and **Finish**.

For Oracle Linux with Chrome:

1. From the browser tab running OMA from the FQDN server, click on the address bar on the "Not Secure" area, and then click on "Certificate". The certificate viewer will popup.
2. In the Certificate Viewer, select the details tab, scroll down to the bottom and click **Export...**, then navigate to a place to save the certificate, on the bottom of the dialog select "DER-encoded binary, single certificate" and click **Save**.
3. From a terminal on the client running Chrome, run the following command to import the certificate:

```
certutil -d sql:$HOME/.pki/nssdb -A -t TCuc -i ~/Downloads/nmshost.oracle.com -name nmshost.oracle.com
```
4. Restart Chrome and reload the OMA, you will now be running OMA as a real PWA.

Client PWA Installation Instructions

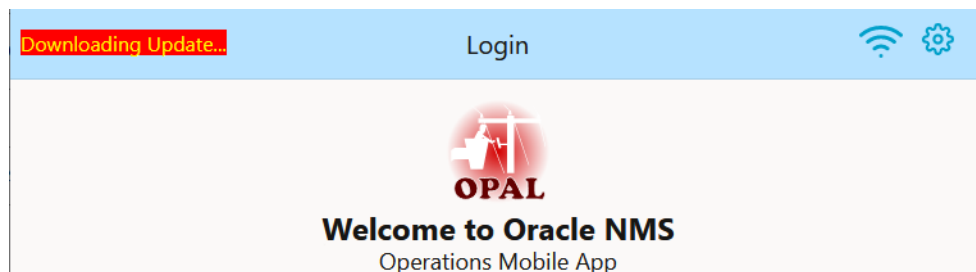
OMA is a PWA application and can be installed on the following types of systems.


Platform	Supported Browsers
Windows 10 and above	Google Chrome Microsoft Edge
Linux	Google Chrome
Android Tablets or Phones	Google Chrome
iOS Tablets or Phones	Safari

OMA installation is done by using the devices browser and connecting to the OMA application website.

OMA has a standardized installation process for Android, iOS, Linux, and Windows:

1. Launch the browser to the HTTPS server where OMA is installed (for example, <https://host.com:8889/mobile/oma2/www/index.html>).
2. On launch, the browser will detect the app is a PWA and will start the download of the PWA to the browser cache storage. This will be identifiable to the user by the flashing "Downloading Update..." in the OMA header bar:



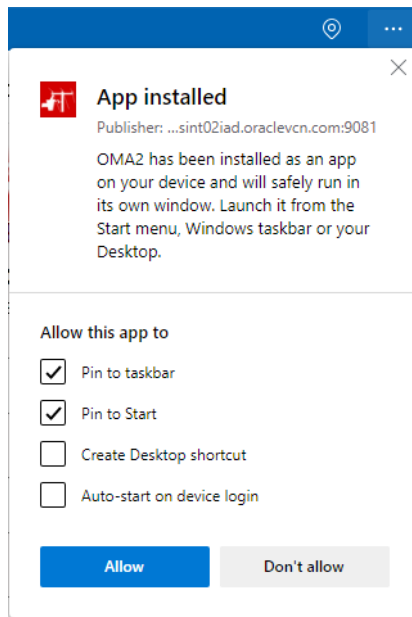
3. On **iOS**, while the PWA is downloading, a dialog screen will be displayed with instructions for installing OMA after the download is complete.
4. Once the PWA download is complete, on all but iOS devices, an **Install as PWA** button will replace the flashing **Downloading Update...** notification.
5. Click the **Install as PWA** button.
6. On iOS systems, once the **Downloading Update...** is complete, navigate to and press the **Share...** button () . Select the option to add OMA to your home screen. Press the OMA icon on the home screen to start the PWA.

7. On Windows with Chrome:

- a. You will be presented with an installation dialog box. Click the **Install** button to install it on the device.
- b. The PWA will close in the browser tab and open as a standalone PWA app.
- c. The PWA will be given a start menu option from where you can launch the app, pin it to the taskbar, pin it to the start menu, or open the location of the app and copy it to the desktop.

8. On Windows with Microsoft Edge:

- a. You will be presented with an installation dialog box. Click the **Install** button to install it on the device.
- b. The PWA will close in the browser tab and open as a standalone PWA app.
- c. You will also be provided a dialog about installation options:



9. On Android Chrome:

- a. You will be presented with an installation dialog. Click the **Install** button to install it on the device.
- b. A small temporary message will indicate the app is being added to the home screen.
- c. Close the browser and the home screen should be focused on the location of the OMA app icon. Feel free to drag it to any location on the home screen(s).
- d. To launch OMA as a PWA, just click on the OMA launch icon.

10. On Linux Chrome:

- a. You will be presented with an installation dialog box. Click the **Install** button to install it on the device.
- b. The PWA will close in the browser tab and open as a standalone PWA app.
- c. The PWA will be given a start menu option, most of the time under section identified as Chrome Apps. From there you most likely will be able to drag it to the desktop.
- d. To launch OMA as a PWA, click the OMA launch icon.

Client PWA Update Instructions

If the client is connected to the network, then on application start-up it will look for a newer version of the PWA application on the HTTPS server. If it finds one, it will start the download and indicate to the user that an update is being downloaded with a flashing notice on the application header, just as was experienced on the initial launch install.

While the download is happening, you may continue to use the application without issues until the download has completed.

When the download does complete, the download notification will be replaced with a notice to click to update the app. You may continue to use the application without issues, the application will not be updated until you restart it or select the install update notice on the header bar.

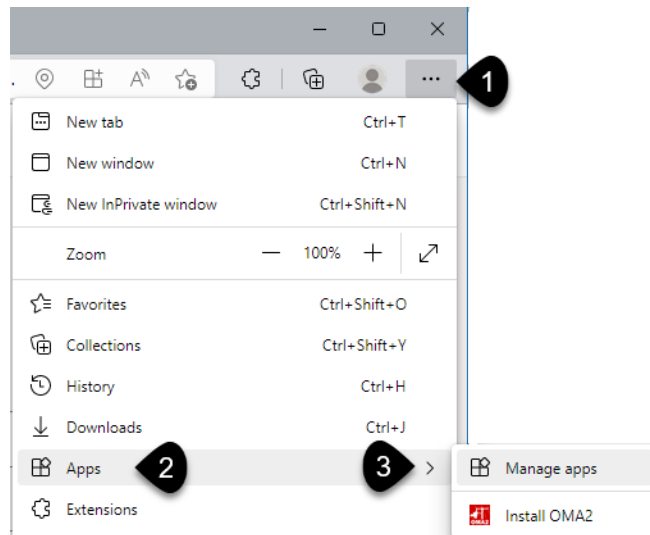
Client PWA Uninstall Instructions

The PWA application removal is different based on the platform.

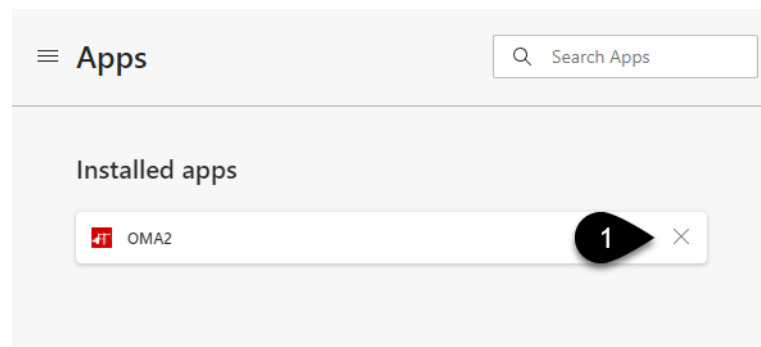
Windows - Edge

There are multiple ways to uninstall the PWA app:

1. From the **Windows Start** menu, open the Settings App. Select Apps and it will go to this page. Scroll down, find the app, select it, and then select **Uninstall**.
2. Open the Edge Browser. From the browser main menu (1), select the **Apps** submenu (2), and then the **Manage apps** (3):



The Edge app manager will open listing the installed apps:



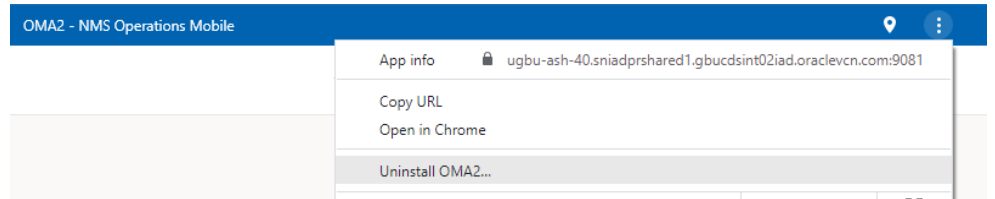
Click the **X** button (1) to open a dialog box with an option to clear data and a **Remove** button to uninstall the app.

3. Open the **Windows Start** menu, right-click the **OMA app** icon, and select **Uninstall**.

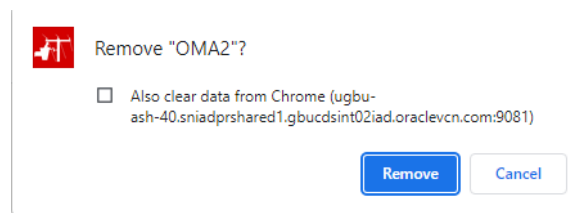
Removing a Chrome OMA PWA on Windows and Linux

Option 1: Windows and Linux

1. From the top of the OMA app, click the three dot menu (⋮).
2. It will open a menu with an option to uninstall the app.



3. When you select that option, a dialog box will open with an option to clear data and a button to remove the app.

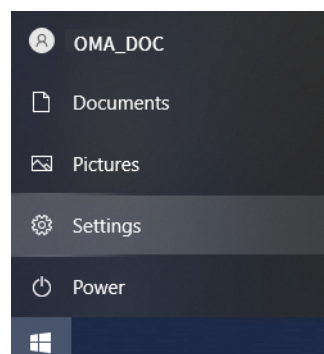


Option 2: Windows and Linux

1. Open Chrome and navigate to Chrome://apps.
2. Right-click the OMA app icon and select **Remove from Chrome**.

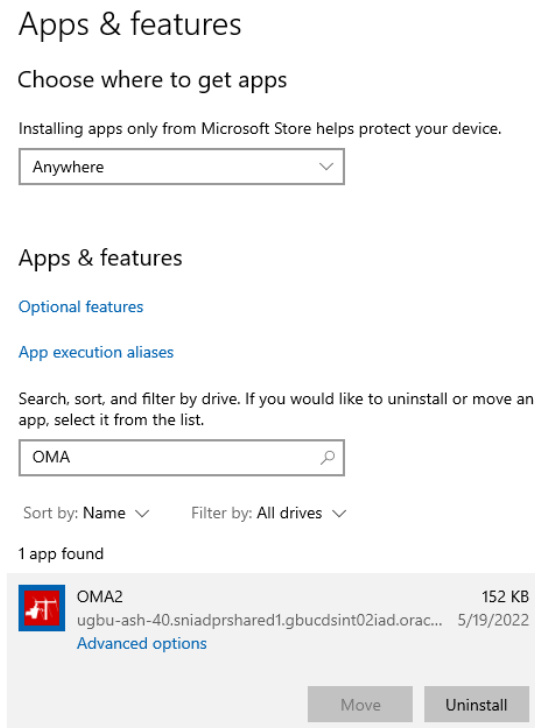
Option 3: Windows

1. From the Windows **Start** menu, click **Settings**:



2. On the **Settings** page, select **Apps**.

3. On the **Apps & features** page, find and select OMA2.



4. Click **Uninstall**. A dialog box will display with a notification:
This app and all related info will be uninstalled.
5. Click **Uninstall** to confirm.

Android

1. From your app drawer or home screen, open **Settings**.
2. Tap Apps & Notifications See all apps.
3. Find the PWA you want to remove and tap the icon.
4. Tap Uninstall.

iOS

PWAs can be deleted the same way any native app is removed.

1. From the Home screen: tap and hold until the icons start jiggling.
2. Select the "x" in the top-left corner of the OMA icon.
3. Lastly, confirm the operation by tapping the **Delete** button.

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` directory. 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.

OMA also supports the Flex configuration override mechanism, which can make it easier to separate project customizations from base product code. To do this you can add configuration overrides in the `.jsonc` files in the `resources/config` directory - there is one for each `.js` file. See the configuration guide for more details of how this override mechanism works.

Migrating Map Configuration from Previous OMA Versions

It is possible to automatically migrate much of the map configuration from older OMA versions by adding the URL parameter `"translateConfig=true"` to the OMA URL when running in a browser during testing. This will add a console log entry with auto generated configuration based on the current configuration and methods in your OMA project. The contents of `resources/config/mapConfig.jsonc` can then be overwritten with this generated configuration. Note that there are some aspects of OMA configuration that cannot be effectively automatically migrated, if changes have been made to some functions in `viewModels/map.js` or `resources/config/map.js`. In particular the `AddLineToElectricBucketStyle`, `redrawConditionLayer`, `isBigSymbolsConditionClassStatus` and various labeling related methods should be compared with product implementation to check for differences, and if there are differences, the changes will have to be re-implemented in the new map configuration

Third Party Application Integration with the OMA Client

OMA supports these methods to allow third party integration:

- [Third Party Application Integration with localhost HTTP Bridge \(Windows 10\)](#)
- [Deep Links Using an OMA PWA Wrapper](#)

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

The OMA Windows 10 browser test applications support a localhost HTTP bridge URI 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, as described in [Configuring OMA Bridge and Wrapper Deep Link Commands](#) below.

Deep Links Using an OMA PWA Wrapper

When OMA is Wrapped using the PWA Wrapper, the wrapper will capture deep link calls and process them in the wrapper or pass them to OMA. OMA will not respond to most deep links unless OMA is already logged on.

The deep link schema is defined in the wrapper and defaults to `nmsoma://`. For example, if a browser or application on the same device as OMA is logged in on makes a URL request to: `nmsoma://mapfocus?lat=40.9253&long=-81.3981`, OMA will be brought to the foreground, the map page will appear, and the map will be focused on the provided lat/long location.

The Wrapper does have a few internal deep link functions it responds directly:

`nmsoma://wrapper/clearCache`: This will cause the Wrapper to clear the cache of the internal WebView.

`nmsoma://wrapper/relaunch`: This will cause the Wrapper relaunch the OMA application in the WebView with the `?runInBrowser=true` parameter.

`nmsoma://wrapper/restart`: This will cause the Wrapper relaunch the OMA application in the WebView with the `?runFromWrapper=true` parameter.

Otherwise all other deep links will be passed to OMA to resolve in the `oma-main.js::handleOpenURL` function and are described in [Configuring OMA Bridge and Wrapper Deep Link Commands](#).

Configuring OMA Bridge and Wrapper Deep Link Commands

OMA will need to be configured to process `/oma/` commands in the `oma-main.js::handleOpenURL` function. The following `/oma/` commands are included in the OMA template code as examples. If using the HTTP bridge, the calls are preceded by `https://localhost:9595/oma/<command>`. If using the wrapper deep links, the calls will be preceded by `nmsoma://`.

All of the commands also support these parameters:

- **uxreturnurl:** If provided, will place an icon in the main OMA header and will use this value as a deep link URL to call when the icon is clicked. Once the icon is clicked, it will disappear.

example:

```
nmsoma://mapfocus?lat=40.9254&long=-81.3983&msg=frisvold&uxreturnurl=http://maps.apple.com/?ll=45.123,-93.505
```

mapfocus

Query Parameters

- **lat:** Latitude to focus the map on (required)
- **long:** Longitude to focus the map on (required)
- **zoom:** Zoom level to focus the map on (optional)
- **msg:** Message to display to the OMA user (defaults to "Map Focused")'
- **forceOpen:** true or false

This call will open OMA to the map page on provided lat/long and zoom level and display the provided message in a toast. If the OMA user is editing something (for example, an event, damage report, and so on). If `forceOpen` parameter is not set to true, OMA will inform the user that a request to focus somewhere else has been received and give them a chance to ignore the focus request or to leave their edits without saving data and honor the focus request.

Example:

```
nmsoma://mapfocus?lat=40.9254&long=-81.3982 or https://localhost:9595/oma/mapfocus?lat=40.9254&long=-81.3982
```

open

Query Parameters

- **page:** "task", "switching" or "safety" (required)
- **id:** ID of item to focus on (either `id` or `event_id` required)
- **event_id:** for switching, will get the first switching sheet associated with the `event_id` (either `id` or `event_id` required)
- **msg:** Message to display to the OMA user (defaults to OMA configured message based on the page type)'
- **forceOpen:** true or false

This call will open OMA to the page to the provided and open the id for that page (for example, if `tasks`, ID will be the task id, if `switching`, will be the switching sheet id, if `safety`, will be the safety doc id) and display the provided message in a toast. For `switching`, if just the `event_id` is provided and not a switching id, OMA will look for the event and see if the event has a switching sheet associated with it, and if so, will open that sheet in the switching page. If the OMA user is editing something (such as an event, damage report, and so on). If `forceOpen` parameter is not set to true, OMA will inform the user that a request to focus somewhere else has been received and give them a chance to ignore the focus request or to leave their edits without saving data and honor the focus request.

Example:

```
nmsoma://open?page=switching&id=1001 or https://localhost:9595/oma/open?page=switching&id=1001
```

mainpage

Query Parameters

- **page:** "task", "switching", "safety", "map", "crew", "profile", ... (required)
- **msg:** Message to display to the OMA user (optional)

This call will open OMA to the main page provided and display the provided message in a toast.

Example:

```
nmsoma://mainpage?page=profile or https://localhost:9595/oma/mainpage?page=profile
```

crewstatus

Query Parameters

- **status:** "ENR", "ONS", "safety", "map", "crew", "profile", ... (required)
- **event:** Event ID to update crew status on (required)
- **suppressNmsCalls:** if 'Y', do not make RestFUL calls to force status

This call will change the crew status for the event ID to the status provided. If `suppressNmsCalls` is set to 'Y', no calls restful service calls will be made to the server to update the crew status, it will be assumed the caller will make that request to NMS. Otherwise OMA will make the calls to update NMS.

Example:

```
nmsoma://crewstatus?status=ENR&event=5023 or https://localhost:9595/oma/crewstatus?status=ENR&event=5023
```

getinfo

Query Parameters

- **type:** "event", ... (required)
- **id:** Event ID to get information for (required)
- **responseurl:** template for the custom URL

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

Example:

```
nmsoma://getinfo?type=event&id=2130&responseurl=http://  
xxx.us?cause={cause} or https://localhost:9595/oma/  
getinfo?type=event&id=2130&responseurl=http://xxx.us?cause={cause}
```

Chapter 6

Client Deployment

Server Application Install Instructions

OMA is deployed as a web application from an https file server. Production deployment the OMA application is only supported via an https file server. Technically, the non-PWA version of OMA will run via an http server, however, for good web application deployment security, only https file servers are supported.

Only one application should be hosted under each https file server (a given host and port) because the applications will share application storage on the client. This includes localStorage, sessionStorage, IndexedDB, WebSQL, and Cache Storage. For example, if you have a production version of OMA on `https://host.com:8080/index.html` and a test version of OMA on `https://host:8080/test/index.html`, they will share the same application storage under the domain `https://host.com:8080`. This is not necessarily an issue, you just need to know if you change a setting that is stored in application storage (for example, Login settings Enable IDCS login or Enable OMAHttpBridge), it will be changed for both version of the application. Another example, if you have two versions of the application, one for your North region and one for the South region and if their are differences more than just the data they interact with, it is recommended to use to different https domains (for example, `https://host.com:8080` for North and `https://host.com:8081` for South).

If you are running the application as a Web app (not a PWA), the https server can be signed with a less secure SSL certificate (for example, self-signed non Certificate Authority (CA)).

Applications running as a Progressive Web App (PWA) use a service worker. When the service worker attempts to register itself, the https server's certificate must be valid. The https server certificate to support a PWA has these requirements:

- The https server must be running on a Fully Qualified Domain Name (FQDN, host.com) that is recognized by the clients DNS server.
- The SSL certificate used by the https server must include the FQDN in the Subject Alternate Name (SAN) list.
- The SSL certificate must be issued by a root level trust authority or CA. On iOS or Android, this must be a client verifiable CA authority. On Windows, you can install a self signed FQDNed certificate with the FQDN in the SAN in the windows system certificate storage using the Manage Computer Certificates app. On Linux systems with a tool called certutil, you can install a self signed FQDNed certificate with the FQDN in the SAN in the Linux system certificate storage for the user.

One more note about PWAs, the RestFUL endpoint the PWA communicates with (for example, the nms-ws RestFUL services) MUST also follow the same certificate requirements of the PWA's https SSL certificate. PWAs can not communicate to untrusted RestFUL service endpoints.

To install the application, copy the www directory from the application build distribution directory to the https server root directory. For example, on a simple OPAL NMS system, there is an nms-lighttpd server serving the \$OPERATIONS_MODELS directory. Follow the instructions in the nms-lighttpd script to enable the https server. For this example, we will assume the host name is host.com and the https port will be 8889. Copy the application build distribution directory OMA2/dist/oma_pwa/www directory to \$OPERATIONS_MODELS/mobile/oma2/www directory. From any client device with access to the nms-lighttpd server, goto the address: https://host.com:8889/mobile/oma2/www/index.html, and OMA will start as a PWA enabled application.

Server Application Update Instructions

To update the OMA application, replace the application files in the same location under the https server root directory.

For web apps, the next time the user restarts their browser and launches the app, they should get the updated application.

For PWA apps, the oma2_build.sh process will update the PWA version and will trigger an update of the client on its next launch if the client has access to the https server.

Chapter 7

OMA Native PWA Wrappers

The OMA Progressive Web Application (PWA) provides a good user experience using common web technologies that can run on any platform supporting standards compliant browsers including PCs, tablets, and mobile devices. However, there are some usability features not yet supported with PWA technology such as running in the background, deep link interaction with third party apps, local device notifications, ability to bring itself to the foreground, ability to install or update using Device Management systems, and more. The OMA Native Wrappers is our solution to fill in the gap and provide these features. Use of OMA Native wrappers is optional and only required if one or more of the above gap items are important to your implementation.

OMA native PWA wrappers are supported on Android and iOS devices. A Windows wrapper is not available currently. The wrappers are small simple native applications built using native development tools requiring no additional third-party software. For the OMA PWA wrapper on Android, you will need an **Android Studio** development environment. For the OMA PWA on iOS, you will need a MacOS system with **XCode** and an Apple Development License.

The OMA PWA Wrappers are released as source code templates (`OMAAndroidWrapper.zip` and `OMAIOSWrapper.zip`) and located in the Oracle Utilities Network Management System Software Development Kit (SDK) `$NMS_BASE/sdk` directory.

PWA wrappers do not contain the OMA application, but instead are built with a configuration pointing to the OMA PWA server. When the wrapper is installed and initially launched, it will go to the OMA PWA server and download the latest version of OMA PWA into the wrapper's storage. Once the OMA PWA is installed in the wrapper's storage, OMA PWA will be able to start and run while the device is off-line. If you have different versions of OMA PWA (for example, *production*, *development*, and *test*), you will need to build a wrapper for each of the locations where OMA PWA is served from. If you use the same OMA PWA to connect to different NMS instances (for example, *production north*, *production west*, and *production south*), you can configure OMA PWA to connect to each of these instances at login time; in other words, there is no need to have a different wrapper for each NMS instance. Once OMA PWA is installed in the wrapper's storage, on each subsequent launch of the wrapper, OMA will check for new versions of the OMA PWA and download them to the device so there is no need to rebuild/reinstall the wrapper to get OMA PWA updates.

Building and Deploying the OMA PWA Wrapper for Android

Follow these steps to build and deploy the wrapper:

1. Install the Android Studio software to your chosen development system. Android Studio is available for Windows, Mac, and Linux systems and can be downloaded from this website: <https://developer.android.com/studio>. The template is using Android Studio Flamingo (2022.1) and Gradle 8.0.
2. Copy the OMA PWA Wrapper for Android from the `$NMS_BASE/sdk/OMAAndroidWrapper.zip` file and unzip it on your development system.
3. Launch Android Studio and open the wrapper project (`OMAAndroidNativeApp`).
4. Configure the following in the wrapper template:
 - a. In the `app/src/main/res/values/strings.xml` file:
 - Update the `app_name`. This is the name the app will have on the device including under the launch icon. If you are only going to have multiple wrappers printing to different OMA PWA servers, this should be unique (for example, *OMA Production* or *OMA Development*).
 - Update the `pwa_address` to reference your OMA PWA server.
 - Update the `wrapper_version`. This will appear on the **About** screens in OMA.
 - Change `channel_name` and `channel_description` to be unique. These are used to setup the local device notification scheme.
 - b. If you want to change the icon for the app used in the launch icon or the notification icon, update the icons in the `app/src/main/res/mipmap` directories. Use instructions in the Android Studio "Create App Icons" section or use one of many online tools (for example, <https://www.appicon.co/>).
 - c. In the `app/src/main/java/MainActivity.java` file:
 - Update the `CHANNEL_ID` and `GROUP_NAME`. These are used to setup the local device notification channel. If you are only going to have multiple wrappers printing to different OMA PWA servers, these should be unique for each wrapper.
 - d. In the `app/src/main/AndroidManifest.xml` file:
 - If you are going to have multiple wrappers installed at the at the same on a device, you will need unique deep link schema names. Change `android:schema` name from `nmsoma` to something unique for each wrapper.
5. Build the wrapper by selecting **Make Project** from the **Build** menu.
6. Build the APK file by selecting **Build Bundle(s)/APK(s)** and then **Build APK(s)** from the **Build** menu. The generated APK (`app-debug.apk`) will be in the `OMAAndroidWrapper\OMAAndroidNativeApp\app\build\outputs\apk\debug` directory; use this to distribute your app. If you want to do more sophisticated APK generation (such as signing your app), refer to the *Android Studio* documentation.
7. To deploy your wrapper, you have many options. Here are some examples, you are not limited to these
 - a. Copy the APK to a web server and share a link to the APK with the users to click on and install on their device. Be sure the web server has the following

```
mimetype.assign including: ".apk" => "application/  
vnd.android.package-archive".
```

- b. Give the APK to your Device Management team to deploy to users.
- c. Plug your Android device into your Android Studio Development environment and tap the Run/Run App menu option with your device selected in the app/device drop-down list on the menu bar. This will install and run the app on your device.

Building and Deploying the OMA PWA Wrapper for iOS

Follow these steps to build and deploy the wrapper:

1. Install the XCode software to your MacOS development system. The template is using MacOS Monterey 12.6.3 and XCode 14.2.
2. Install your Apple Developer Credentials and provisioning files in XCode. Refer to Apple Developer and XCode documentation.
3. Copy the OMA PWA Wrapper for iOS from the `$NMS_BASE/sdk/OMAIOSWrapper.zip` file and unzip it on your development system.
4. Launch XCode and open the wrapper project by selecting **Open...** from the **File** menu, navigating to the unzip location, and opening the `OMA_IOS_Wrapper.xcodeproj` XCode project.
5. Configure the following in the wrapper template:
 - a. In the Info file:
 - Expand the WKAppBoundDomains. Change the "item 0" value from `oraclevcn.com` to the domain name where your OMA PWA is served from. If the OMA PWA is served from `https://systema.mouse.cat.animals.com`, you need to at least include `animals.com` in this value. Refer to the Apple Developers documentation for more details.
 - Expand the URL Types. If you would like to change the deep link used by OMA from `"nmsoma://"` to something else.. change the URL Identifier value and the URL Schemas/Item 0 value to what you want to use.
 - b. **Assets:** If you would like to change the icons used by the wrapper (launch icon, notices icon, ...), update the Assets/AppIcons.

Refer to the Apple Developer documentation for more details. There are many free icon generator tools on line to help generate all the variants needed (<https://www.appicon.co/>).
 - c. **ViewController:** There are two locations where the OMA PWA URL needs to be specified, replace the template URL (`https://ugbu...`) with your URL. Be sure you keep the `?runInBrowser=true` and `?runFromWrapper=true` from the template URL.
6. Build the wrapper by selecting **Build** from the **Product** menu.
7. Build the ipa file:
 - a. Select **Archive** from the **Product** menu.
 - b. When the archiving is complete, an archives list will be displayed. From the list, select the latest archive and then click the **Distribute App** button. Select the method of distribution matching your Apple Developers License level

(**Enterprise**), then click **Next**, and, on the distribution options panel, set **App Thinning** to **None**.

If you would like to use over-the-air method of installation, select the corresponding box, click **Next**, then fill in the over-the-air parameters for your installation website page, and click **Next**.

If you are not using the over-the-air method of installation, do not select the box and click **Next**. On the next panel, enter the signing method you would like to use and finish the rest of the dialog boxes. Finally, export the ipa package using the export button when it appears.

For more details, refer to the Apple Developers documentation.

8. You have many options to deploy your wrapper. Some examples include:
 - a. Copy the ipa file and over-the-air configuration to a web server as specified when creating the export. Users can then install the app by navigating to that URL.
 - b. Give the ipa file to your Device Management team to deploy to the users.
 - c. Plug your iOS device into your XCode Development environment and select **Run** from the **Product** menu to install and run the app on your device.

How to Issue a Local Notification from OMA When Using the PWA Wrapper

The OMA wrappers (iOS and Android) both support local notification functionality using the `window.Android.notification` API:

```
window.Android.notification ( id, title, text[, tap-deep-link])
```

where:

- **id:** index number to be used for the notification. For OMA, the `store.js` has a function to generate these: `getNotificationId()`
- **title:** Text string to use for the notification title
- **text:** Body text to use for the notification, supports `\n`'s
- **tap-deep-link:** An optional deep link to call if the user taps the notification. For example, `'nmsoma://mainpage?page=tasks'` would focus OMA on the task list page.

Chapter 8

Operations Mobile Application Setup on OPAL

The OPAL demonstration project has been configured to be Operations Mobile Application ready. To get the Operations Mobile Application running on an existing running OPAL 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 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. 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,
};
```

6. Build the Operations Mobile Application install file for each targeted platform/device.
7. Install the Operations Model Application on each targeted device.

-
8. Deploy the Mobile Gateway on the WebLogic Application Server.
 9. Test the Operations Mobile Application.

Chapter 9

Operations Mobile Application Project Setup

To configure the Operations Mobile Application on your 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 -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. 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,
};
```

7. In the development environment, configure your device, condition, and conductor symbology. Put the device and condition `.svg` files in the `src/js/symbols`

directory. Map the files to the geojson object SYMBOL using the FLEX configuration system as described in the `OMA2/src/ts/client/config/examples/README.txt`.

8. 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";
```

9. 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',
  ...
}
```

10. 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'
  ...
}
```

11. 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',
  ...
}
```

12. 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"]
        }
      ]
    }
  ]
}
```

```
    ],  
    ...
```

13. 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  
},  
    }  
    ...
```

14. In the development environment, configure your map object attribute viewer options as defined in **Configuring OMA Object Attribute Viewer** on page 4-10.
15. 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'},  
    ];
```

16. In the development environment, configure your map object Search options as defined in **Configuring OMA Search Options** on page 4-13.
17. 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'},  
    ];
```

18. In the development environment, build the Operations Mobile Application.
19. 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`.
20. Configure your Mobile Application Keys in the NMS Configuration Assistant or project scripts.
21. Install the Operations Model Application on each targeted device.
22. Deploy the Mobile Gateway on the WebLogic Application Server.
23. 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.