# FY14 MAW Phone App – Programmer’s Overview

## Introduction

The purpose of this document is to provide a high-level overview of the MAW phone app and underlying code. It is intended for readers with extensive experience with the technologies employed in the development and deployment of the application, such as java, the Android Development Kit, Eclipse IDE, multi-threading, and mobile application development.

Nota bene: There have been reports that the app continues running between user sessions, and may be responsible for excessive battery drain on the device. This is most likely due to the GPS, which may continue to determine the device location. In order to prevent battery drain after you are done using the app, go to the Android Application Manager (on some phones by holding down the Home button) and force MAW to quit.

## Application Description

The MAW phone app is a java application intended to demonstrate the capabilities of the VDT system developed under TOPR4. The app retrieves an alert for the current location and displays text for the alert on the screen of the device. The app may also play an audio alert if it is necessary. The app communicates with a services layer that performs all of the alert computations.

The display is designed to run on an android device running version 4.1.2 or later of the operating system. The display assumes that GPS location is available from the device, and that there is a reliable connection to the internet. Best effort was made with available development resources to handle error cases, but the user may experience undesirable behavior when there are problems with internet connectivity or determining the location of the device.

The phone app performs the following functions in the course of a typical user session:

1. **Begin Session**  
   When the user toggles the slider to turn alerts on, the app contacts the service layer to create a new session. Once the session is initiated, the app starts a timer that requests an alert every 15 seconds. To change the interval between alert updates, change the value in the loginComplete() method of the Controller class. The value has units of milliseconds.
2. **Determine Location**  
   Android provides utilities to determine the device location. The MAW phone app registers for location updates when alerts are turned on. The latest location update is cached on the Controller class for use by the app.
3. **Request Alert**  
   Every 15 seconds the app sends the latest location to the service layer and requests the most recent alert for that location. The resulting alert is used to update the display.
4. **Change Screen Color**When a new alert is received, the alert action code is used to set the screen color. See the Lookup class for the code that defines this behavior.
5. **Display Text Alert**  
   Alert text is displayed based on the four alert codes associated with the latest alert. The text is concatenated from string resources associated with each of the alert codes. See the alertUpdateComplete() method of the Controller class for the code that assembles the alert text.
6. **Play Audio Alert**  
   If the new alert is more severe than the previous alert, or if no alert has been played for 15 minutes, an audio alert is played on the device. The audio alert corresponds to the text alert displayed on the screen, and is comprised of one or more mp3 files played in sequence. See the alertUpdateComplete() method of the Controller class for the code that chooses the mp3 files to play, and the logic for when they are played.
7. **Send Confirmation**  
   Following update of the display and possibly playing the audio alert, the display sends a confirmation to the services layer indicating that the alert has been received and used to update the display.
8. **End Session**  
   When the user toggles the alert slider to the off position, the display sends a logout request to the server to terminate the session.

## Application Architecture

The MAW phone app is a java application compiled for deployment to an android device. The application architecture is largely determined by conventions inherent with android development. The primary code and configuration components are listed here in order to provide some guidance to developers.

### MainActivity

The MAW phone app contains two activity classes – the main activity and the settings activity. MainActivity handles app initialization and all the callbacks for UI events. Any behavior associated with these callbacks is delegated to the Controller class.

### Controller

The Controller class contains most of the behavioral logic of the app. This class initiates all the client-server interactions, and updates the UI when new alerts are received. It also plays the audio alerts when appropriate. All of the client-server interactions are asynchronous – called on their own threads created as subclasses to android.os.AsyncTask – so any behavior necessary after a server request is handled by Controller calls from the onPostExecute() method of each Task.

### Settings

Application settings, or user preferences, are handled by the SettingsActivity class. This class manages the UI for displaying and changing user preferences, and takes care of storing the settings between uses of the app.

### Alert Configuration

Text and audio alerts are stored as Android resources in the res/ directories. Text alerts are stored as strings in res/values/strings.xml and audio alerts are stored as raw mp3 files in res/raw. The values and files are looked up based on the alert codes provided by the service layer. Changes to existing alerts can be accomplished by modifying the resources themselves. Adding new alerts will require editing the Lookup class – add the new alert codes and resources to the existing alerts by following the simple patterns present in each of the methods.

### Client-Server Interaction

The MAW phone app uses a REST-like protocol to request data from the services layer. Data is returned in JSON format. Below are example requests and responses from each of the services:

**Login**  
Service Name: maw\_login\_dev  
  
URL: http://www.ral.ucar.edu/projects/rdwx\_mdss/proxy.php?path=/maw\_login\_dev&phone\_id=3034972821&state=minnesota&action=login  
  
JSON Response:

{

"sequence\_no": 1,

"session\_id": 1393496926

}

**Logout**  
Service Name: maw\_login\_dev  
  
URL: http://www.ral.ucar.edu/projects/rdwx\_mdss/proxy.php?path=/maw\_login\_dev&session\_id=1393496926&phone\_id=3034972821&state=minnesota&action=logout  
  
JSON Response:

{

"sequence\_no": -9999,

"session\_id": 1393496926

}

**Alert**  
Service Name: maw\_alerts\_dev   
  
URL: http://www.ral.ucar.edu/projects/rdwx\_mdss/proxy.php?path=/maw\_alerts\_dev&phone\_id=3034972821&lat=40.1&lon=-92.221&session\_id=1393496926&prev\_sequence\_no=1&state=minnesota  
  
JSON Response:

{

"alert\_request\_time": "201402271030",

"sequence\_no": "2",

"alert\_code\_precip": 0,

"alert\_action\_code": 0,

"session\_id": "1393496926",

"alert\_time": "20140227103800",

"alert\_code\_pavement": 0,

"alert\_code\_visibility": 0,

"alert\_gen\_time": "20140227103800"

}

**Alert Confirmation**  
Service Name: maw\_alerts\_confirm  
  
URL: http://www.ral.ucar.edu/projects/rdwx\_mdss/proxy.php?path=/maw\_alerts\_confirm&session\_id=1393496926&prev\_sequence\_no=1&played\_audio=t

JSON Response:

{

"confirmed\_sequence\_no": "2",

"response": "OK",

"session\_id": "1393496926"

}

## Development Environment

The MAW phone app was developed on a Mac with Eclipse and the Adroid SDK:

* Platform: Apple MacbookPro 2.53 GHz Core i5 with 8GB RAM
* OS: OS X 10.8.5
* IDE: Eclipse 4.2.2 Build id: M20130204-1200 with the following software:
  + Android DDMS 22.3.0
  + Android Development Tools 22.3.0
  + Android Hierarchy Viewer 22.3.0
  + Android Native Development Tools 22.3.0
  + Android Traceview 22.3.0
* Android SDK: With latest updates (through API 19)
* Test Environment: Android Emulator running Android version 4.1.2 and Galaxy Note II running Android version 4.3.0