**Pikalert® Vehicle Data Translator 5.0**

**Enhanced Maintenance Decision Support System**

**Motorist Advisory and Warning**

**Application Installation Guides  
  
Version 1.1**

**March 2017**



image courtesy iStockPhoto (<http://www.istockphoto.com/stock-photo-11156851-rush-hour-traffic-in-the-rain.php>)

**Prepared for:**

**Federal Highway Administration**

**(FHWA)**

**Road Weather Management Program**

**Prepared by:**

**The National Center for Atmospheric Research**

**RELEASE NOTES**

|  |  |  |
| --- | --- | --- |
| **Version Number** | **Date** | **Notes** |
|  |  |  |
| Version 1.0 | 14 December 2016 | Incorporates new enhancements |
| Version 1.1 | 30 March 2017 | Incorporates additional edits |
|  |  |  |

Questions and comments about this document should be directed to:

Project Oversight and Engineering: Science:

Dr. Gerry Wiener Amanda Anderson

Research Applications Laboratory Research Applications Laboratory

NCAR NCAR

P.O. Box 3000 P.O. Box 3000

Boulder, CO 80307 Boulder, CO 80307

Ph: 303-497-8417 Ph: 303-497-2746

E-mail: [gerry@ucar.edu](mailto:myers@ucar.edu) E-mail: [aander@ucar.edu](mailto:gerry@ucar.edu)

**TABLE OF CONTENTS**

1 INTRODUCTION 13

2 INTENDED AUDIENCE 13

3 OVERVIEW 13

3.1 Disclaimer 14

3.2 Pikalert VDT Improvements since Version 4.0 14

Science Enhancements 14

Engineering Enhancements 14

3.3 Summary of Pikalert VDT Improvements since Version 3.0 14

4 RELATED DOCUMENTS 16

5 INSTALLATION INSTRUCTIONS 16

5.1 Pikalert VDT and RWH 17

5.2 Installing Pikalert RWFS 20

5.3 Installing Pikalert RCTM 22

5.4 Installing the EMDSS and MAW Back End Server Software 25

6 PIKALERT VDT, EMDSS, AND MAW SYSTEMS OVERVIEW 28

6.1 System Goals 28

6.2 System Structure 29

6.2.1 Computer Hardware 29

6.2.2 Software Architecture 29

6.2.3 Computer Languages 30

6.2.4 External Software Requirements 30

6.2.5 Data Ingest 31

6.3 Application Components Overview 31

6.3.1 Weather Data Ingest 32

6.3.2 Vehicle Data Translator 33

6.3.3 Road Weather Forecast System Diagram 33

6.3.4 Road Condition and Treatment Module 34

6.3.5 Road Weather Hazard Module 35

6.3.6 Road Weather Alert Module 35

6.3.7 EMDSS Module 36

6.3.8 MAW and MAW Phone Application Modules 37

7 SYSTEM OPERATIONS 37

8 CONFIGURATION FILES 38

8.1 Configuration File Overview 38

8.2 Road Segment Files 38

8.3 VDT Configuration File 39

8.4 RWH Configuration File 40

8.5 RWA Configuration File 40

8.6 Site List Configuration Files 41

8.7 Back end server configuration files 42

9 DATA INGEST 42

**Acronym Glossary**

DICast – Dynamic Integrated foreCast

DOT – Department of Transportation

EMDSS – Enhanced Maintenance Decision Support System

FHWA – Federal Highway Administration

HDF – Hierarchical Data Format

IMO – Integrated Mobile Observations

MAW – Motorist Advisory and Warning

MDSS – Maintenance Decision Support System

METAR – Meteorological Terminal Air Report/Aviation Routine Weather Report

METRo – Model of the Environment and Temperature of Roads

NetCDF – Network Common Data Format

NCAR – National Center for Atmospheric Research

OS – Operating System

QCh – Quality Checking

RCTM – Road Condition and Treatment Module

RWA – Road Weather Alert

RWFS – Road Weather Forecast System

RWH – Road Weather Hazard

RWIS – Road Weather Information System

UCAR – University Corporation for Atmospheric Research

USDOT – United States Department of Transportation

VDT – Vehicle Data Translator

# 1 INTRODUCTION

This document describes how to install the Pikalert®[[1]](#footnote-1) Vehicle Data Translator (VDT) Version 5, Enhanced Maintenance Decision Support System (EMDSS), and Motorist Advisory and Warning (MAW) applications. This document is organized as follows:

* Brief overview of the applications’ history and purpose
* List of related documents for more detailed information on the systems and associated research projects
* Installation instructions for the Pikalert VDT
* Overview of the systems, including system structure, third-party software requirements, and application components overview
* System operations
* Configuration files and some examples

# 2 INTENDED AUDIENCE

This document is intended for software engineers or system administrators interested in installing the Pikalert software along with the EMDSS and MAW applications. Experience with the Linux Operating System (OS) and package manager and programming languages such as C++ and Python, as well as knowledge of standard meteorological data sets will be helpful in understanding this document.

Note that the system testing was performed on the Debian Wheezy Linux OS distribution. This is the recommended OS distribution for running the system, but it should work with other Linux distributions such as Red Hat.

# 3 OVERVIEW

This Integrated Mobile Observations (IMO) project was organized by the U.S. Department of Transportation’s (USDOT) Federal Highway Administration (FHWA) in collaboration with the Nevada, Minnesota, and Michigan State DOTs and the National Center for Atmospheric Research (NCAR) to demonstrate how weather, road condition, and other related vehicle data may be collected, transmitted, processed, and used for decision support applications and activities. To that end, the Pikalert VDT was enhanced and two applications, the EMDSS and MAW, were developed using its output.

It is envisioned that components of the EMDSS and MAW applications developed by this project will be further developed, integrated with other operational components, and deployed by road operating agencies, including state DOTs, and generally supplied by the private sector.

## 3.1 Disclaimer

The Pikalert VDT, EMDSS, and MAW described herein are prototype software systems. For this reason, the software should be used with caution. It is anticipated that the prototype software will be used as a springboard toward the development of commercial road weather systems that contain Pikalert, EMDSS, and/or MAW features and functions. How the materials that make up this release are used to create an operational capability is ultimately up to the private sector firms or other organizations seeking to provide those services. Some may choose to utilize the prototype code while others may use the contents as general guidelines for their own development process. It is hoped that the code and capabilities described herein and delivered will accelerate the time to market for operational Pikalert, EMDSS, and MAW technologies.

## 3.2 Pikalert VDT Improvements since Version 4.0

A number of science and engineering enhancements and improvements were made to the Pikalert System to promote diagnosis and forecast accuracy as well as to improve the capability of the Pikalert display. The enhancements/improvements are as follows:

#### Science Enhancements

* Ingest and utilize dual-polarization radar data in Pikalert
* Make improvements to the road slickness algorithm
* Improve the performance of precipitation inference in the Road Weather Hazard algorithm

#### Engineering Enhancements

* Incorporate Weather Telematics data in Pikalert
* Correct radar data timing on the Pikalert display
* Avoid flickering of road weather alerts
* Configure new road segments and incorporate RWIS road state fields
* Improve Pikalert display usability

These enhancements are described in more detail in “Pikalert VDT Enhancements, Operations, & Maintenance Final Report.”

## 3.3 Summary of Pikalert VDT Improvements since Version 3.0

The VDT has been refined since Version 3.0 (Mar 2013) to improve the Quality Checking (QCh) tests and the Road Weather Hazard (RWH) module, which outputs current hazards along the roadway using information from the VDT road segment statistics and is part of the Pikalert system.

The following QCh updates were made:

* Spatial Test filter was reduced from 69 miles to a more reasonable 50 km (31 miles) radius

The following Road Weather Forecast System Module (RWFS), which produces the forecasts used by EMDSS and MAW, updates were made:

* The gridded forecast engine was switched from DICast to Logicast
* An application was written to extract weather forecast data from all the configured road segments from the Logicast grid

The following RWH updates were made:

* The RWH was made a separate module from the QCh and statistics-generating capabilities of the VDT itself
* Algorithm structures updated based on data gathered during VDT 3.0 operations and lessons learned from performance of VDT 3.0
* Logic updated to allow outputs on segments with missing/insufficient mobile data
* Logic updated to allow generalized outputs with insufficient ancillary data
* Added confidence value to outputs to reflect degree to which data is insufficient/uncertain
* Produces additional inferences based on forecasted inputs from RWFS
* Precipitation algorithm output feeds into road condition and visibility algorithms rather than repeating logic steps in those algorithms
* Road splash output type transferred from pavement condition algorithm to precipitation algorithm
* Hydroplaning and black ice outputs added
* Road temperature input added

Modules that were added to support the EMDSS and MAW application developments were:

* Road Weather Alert (RWA) Module – Collects data from the RWH and creates alerts on road segment based on user-defined rankings that can then be sent to the EMDSS and MAW
* EMDSS and MAW Back-end Modules – Two separate modules collect data needed by the EMDSS and MAW applications and serves this data to the display and phone applications
* EMDSS and MAW Display Modules – Two separate modules display data from the back-end in an end user-relevant way
* MAW Phone Application – Provides relevant data from the back-end to the end-user with a text-to-voice feature for use during travel
* MAW Evaluation Module – Collects data when alerts are issued to the mobile phone application and interacts with the driver to get user-based evaluations

# 4 RELATED DOCUMENTS

For additional information on the Pikalert, EMDSS, and MAW applications and the IMO project, the reader is directed to related project documents listed in Table 1.

Table 1. Related Documents

|  |  |
| --- | --- |
| **Document and/or Web Sites** | **Source** |
| Results from the Integrated Mobile Observations Study  <http://ntl.bts.gov/lib/48000/48300/48314/Final_Report_Task_5b_5-31-131.pdf>  FHWA Road Weather Management Program Integrated Mobile Observations (IMO) Project  <https://ams.confex.com/ams/95Annual/webprogram/Paper270297.html> | Federal Highway Administration |
| Integrated Mobile Observations Enhanced MDSS and Motorist Advisory and Warning Applications System Design Description  The Enhanced Maintenance and Decision Support System: User Guide  The Motorist Advisory and Warning System: User Guide | National Center for Atmospheric Research |
| Maintenance Decision Support System (MDSS) Web Site:  <http://www.rap.ucar.edu/projects/rdwx_mdss/>  MDSS Document website:  <http://www.rap.ucar.edu/projects/rdwx_mdss/documents/> | National Center for Atmospheric Research |
| Connected Vehicles website:  <http://rap.ucar.edu/projects/connected_vehicles/>  Connected Vehicles Documents website:  <http://rap.ucar.edu/projects/connected_vehicles/publications.php> |  |

# 5 INSTALLATION INSTRUCTIONS

The system distribution contains the latest installation instructions. They are located in the pikalert/build\_install subdirectory. In particular, see the multiple INSTALL and README files located there. For example, build\_install currently contains the following INSTALL and README files:

INSTALL

INSTALL.backend\_server

INSTALL.rctm

INSTALL.rwfs

README

README.backend\_server

README.rctm

README.rwfs

There are additional files in the distribution that discuss the EMDSS and MAW display and phone applications and their installation.

EMDSS\_Installation.docs

EMDSS\_Overview.docx

MAW\_Desktop\_Installation.docx

MAW\_Desktop\_Overview.docx

MAW\_Phone\_Installation.docx

MAW\_Phone\_Overview.docx

## 5.1 Pikalert VDT and RWH

The following steps describe how to install the Pikalert VDT, RWH software. Note that the INSTALLATION STEPS below are also contained in pikalert/build\_install/INSTALL.

INSTALLATION STEPS

1. See Section 6 for hardware and operating system requirements. Be sure to have installed all third party software listed in Section 6.2.4 before proceeding. Note that the third party software packages can be installed using a package manager or can be built manually.
2. Create User Account
   1. It is recommended that you create a separate user account for running the VDT. This will allow more control over settings and permissions and will simplify removal.
3. Set Environment Variables
   1. VDT\_Root and other environment variables are located in build\_install/pikalert\_env\_vars.csh.
   2. Set VDT\_ROOT to the base directory where the VDT will be installed.
   3. Set RAL\_INC\_DIR and RAL\_LIB\_DIR to the local system’s include and library directories.
   4. Set NETCDF\_ROOT\_DIR, HDF\_ROOT\_DIR, ZLIB\_ROOT\_DIR, SZIP paths, and udunits2 paths to the local system’s netCDF, HDF, zlib, szip, and udunits2 installation directories. These should have been installed as part of step 0.
   5. Once all environment variables have been set, run the following command to export them: source pikalert\_env\_vars.csh
4. Compile the code by executing the following command. This will build all VDT libraries and executables. The libraries will be installed in $VDT\_ROOT/lib and the executables will be installed in $VDT\_ROOT/bin:

./build.py –s manifest .. $VDT\_CODE $VDT\_DATA

1. Check the binaries by running the commands in $VDT\_ROOT/bin without arguments to be sure they print out a usage message.   
     
   Note that you may have to set the LD\_LIBRARY\_PATH environment variable so that executables can find dynamic libraries such as libnetcdf. This is an example but the path may be different on your system:

setenv LD\_LIBRARY\_PATH /home/vii/lib:/usr/local/netcdf/lib: /usr/local/hdf5/lib:/usr/local/python/lib:/usr/local/lib:/usr/dt/lib:/usr/lib:/usr/ccs/lib:/d2/local/lib:/d1/local/lib:/usr/local/szip/lib

1. Configure the system.

Some of the configuration files have hard-coded paths that you may need to modify for your system. Below is a list of the field and the files that may need modification:

1. "rwh\_cdl" in rwh\_config.cfg
2. "road\_segment\_file" in the rwa configuration files (mi\_rwa.cfg, mn\_rwa.cfg, nv\_rwa.cfg)
3. "climatology\_file", "probe\_message\_cdl", "probe\_message\_qc\_cdl", "statistics\_cdl", "assessment\_cdl" in statistics\_output.cfg, vdt\_mi\_config.cfg, vdt\_mn\_config.cfg, vdt\_nv\_config.cfg
4. various paths in backend\_sys\_path.py
5. "RWIS\_DIR" in vii\_paths.py
6. "cdl\_file" in write\_probe\_message\_cdf.py
7. Modify log paths (/d2/vii/data/log) in run\_cdf\_to\_csv.py, run\_create\_probe\_msg\_history\_mi.py, run\_create\_probe\_msg\_history\_nv.py, run\_create\_probe\_msg\_history.py, run\_group\_umtri\_probe\_message\_files.py

To uninstall Pikalert, simply remove the files created during this installation process in install directories $VDT\_CODE and $VDT\_DATA.

## 5.2 Installing Pikalert RWFS

The Road Weather Forecast System (RWFS) is a simplified forecast system

that uses freely-available Numerical Weather Prediction (NWP) model data to

produce data in a format that can be used by the Road Condition Treatment

Module (RCTM).

The RWFS requires only NWP data to run. The software distribution contains all the software and scripts to run it and sample data to test the build.

The distribution is configured to make RWFS forecasts for sites in Michigan,

Minnesota and Nevada from February 2015. Sample input data files are located in test/data/rwfs/. These sample files are used for testing the build.

Before proceeding, note that the information below is also contained in pikalert/build\_install/INSTALL.rwfs.

The Pikalert RWFS is dependent on the following third party libraries and executables:

NetCDF-C

netcdf-4.2.1.1.tar.gz

Available for download at: http://www.unidata.ucar.edu/downloads/netcdf/index.jsp

NetCDF-3 C++

netcdf-cxx4.2.tar.gz

Available for download at: http://www.unidata.ucar.edu/downloads/netcdf/index.jsp

udunits2

udunits-2.1.24.tar.gz

Available for download at: https://www.unidata.ucar.edu/downloads/udunits/index.jsp

HDF5

hdf5-1.8.10-patch1.tar.gz

Available for download at: ftp://ftp.hdfgroup.org/HDF5/current/src/

Jasper

version 1.x

Available as a linux package

PNG

version 12.x

Available as a linux package

zlib

zlib-1.2.5.tar.gz

Available for download at: http://zlib.net/

or http://sourceforge.net/projects/libpng/files/zlib/1.2.5/zlib-1.2.5.tar.gz/

Python-2.7.1

Python-2.7.1.tgz

Available for download at: http://www.python.org/download/releases/2.7.1/

(Note that Anaconda Python can be used: https://store.continuum.io/cshop/anaconda/)

Scons

scons-2.3.0.tar.gz

Available for download at: http://www.scons.org/download.php

Ncftp

ncftp-3.2.5

Available for download at: http://www.ncftp.com/download

INSTALLATION STEPS

1) Install 3rd party packages listed above

A package manager can be used to grab these. This is done as root.

2) Create user account

It is recommended that you create group account to run the RWFS. This will

allow more control over settings and permissions and will simplify removal.

3) Set environment variables

Set the environment variables to the desired paths, and export them using

the following command. (Assumes you are using a tcsh or csh shell.)

cd to where this file (INSTALL.rwfs) is located

emacs/vi rwfs\_env\_vars.csh

source rwfs\_env\_vars.csh

4) Compile and install the code

Run the script below:

./build\_rwfs.csh

This will build all libraries and executables. Libraries will be installed

in $LOCAL\_LIB\_DIR, executables in $LOCAL\_BIN\_DIR, scripts will be copied

to $LOCAL\_SCRIPTS\_DIR.

5) Test the build

Run the script below:

./test\_rwfs.csh

Refer to the output of the test script to fix any errors.

The output of the RWFS is a netCDF file which can be used as input to the

RCTM or other Pikalert components. After the above test runs successfully, a

sample file will be located in $RWFS\_ROOT\_DIR/derive\_rdwx/<date> directory.

6) Further steps

A sample crontab file (crontab.rwfs) is provided in the scripts/crontab

directory which shows how the RWFS components are run in a chain every hour

in real-time. The RWFS requires weather model data to run in real-time.

This release contains scripts and applications to ftp and decode NAM model

data from NCEP for this purpose. The test above uses a sample of this data,

located under test/data/rwfs.

## 5.3 Installing Pikalert RCTM

The Road Condition and Treatment Module (RCTM) is a collection of software that has been developed over many years by a number of organizations, including:

NCAR/RAL: National Center for Atmospheric Research/Research Applications Lab

MIT/LL: Massachusetts Institute of Technology/Lincoln Labs

EC/CMC: Environment Canada/Canadian Meteorological Center

This module is used by the FHWA Maintenance Decision Support System (MDSS), the Enhanced MDSS (EMDSS) and Pikalert systems.

Given roadway information at specific locations, a weather forecast and recent observations at those locations, the RCTM will produce a road temperature forecast as well as provide treatment recommendations for the locations.

The software distribution contains all the software that composes the RCTM, scripts to

run it and sample data to test the build.

Note that the RCTM requires an input weather forecast and recent observations at forecast sites. This distribution contains sample datasets covering roadways in Michigan, Minnesota and Nevada from February 2015. The sample netCDF data files are located in test/data/rctm/. These sample files are used for testing the software build and also provide an example of the data input requirements if a user chooses to run the RCTM in real-time mode.

Before proceeding, note that the information below is also contained in pikalert/build\_install/INSTALL.rctm.

The Pikalert RCTM is dependent on the following third party libraries and build software:

NetCDF-C

netcdf-4.2.1.1.tar.gz

Available for download at: http://www.unidata.ucar.edu/downloads/netcdf/index.jsp

NetCDF-3 C++

netcdf-cxx4.2.tar.gz

Available for download at: http://www.unidata.ucar.edu/downloads/netcdf/index.jsp

HDF5

hdf5-1.8.10-patch1.tar.gz

Available for download at: ftp://ftp.hdfgroup.org/HDF5/current/src/

zlib

zlib-1.2.5.tar.gz

Available for download at: http://zlib.net/

Or http://sourceforge.net/projects/libpng/files/zlib/1.2.5/zlib-1.2.5.tar.gz/

Python-2.7.1

Python-2.7.1.tgz

Available for download at: <http://www.python.org/download/releases/2.7.1/>

(Note that Anaconda Python can be used: https://store.continuum.io/cshop/anaconda/)

Scons

scons-2.3.0.tar.gz

Available for download at: <http://www.scons.org/download.php>

INSTALLATION STEPS

1) Install 3rd party packages listed above

A Debian package manager can be used to install these. This should be done as the root user.

2) Create user account

It is recommended that you create a group account to run the RCTM. This will

allow more control over settings and permissions and will simplify removal.

3) Set environment variables

Set the environment variables to the desired paths, and export them using

the following command. (Assumes you are using a tcsh or csh shell.)

cd to where this file (INSTALL.rctm) is located

emacs/vi rctm\_env\_vars.csh

source rctm\_env\_vars.csh

4) Compile and install the code

Run the script below:

./build\_rctm.csh

This will build all libraries and executables. Libraries will be installed

in $LOCAL\_LIB\_DIR, executables in $LOCAL\_BIN\_DIR, scripts will be copied

to $LOCAL\_SCRIPTS\_DIR.

5) Test the build

Run the script below:

./test\_rctm.csh

Refer to the output of the test script to fix any errors.

The output of the RCTM are netCDF files which contain road temperature

forecasts. After the above test runs successfully, sample files will be

located in $RCTM\_ROOT\_DIR. Each type of file corresponds to a certain

treatment scenario:

merge\_no\_tmt - no treatment; road temps are computed assuming snow remains

on the road

merge\_cur\_tmt - current treatment; road temps are computed using the current

treatment plan

merge\_rec\_tmt - recommended treatment; road temps are computed using the

internal treatment recommendation which removes snow

6) Further steps

A sample crontab file (crontab.rctm) is provided in the scripts/crontab

directory which shows how the RCTM components are run in a chain every hour

in real-time. The RCTM requires weather forecast and observation input files

to run in real-time. Sample netCDF input data files are located under

test/data/rctm. Weather forecast data can be obtained in real-time by

installing the RWFS, and observation data can be obtained from MADIS files.

## 5.4 Installing the EMDSS and MAW Back End Server Software

The backend server consists of a number of Python wsgi modules and Python scripts that support the EMDSS and MAW web displays as well as the MAW phone application.

DEPENDENCIES

The following third party libraries and executables are required for the back end server:

Python-2.7.1

Python-2.7.1.tgz

Available for download at: http://www.python.org/download/releases/2.7.1/

You can also install Anaconda Python. It contains many of the python third party libraries mentioned below.

pytz

pytz-2013.9-py2.7.egg

Available for download at: https://pypi.python.org/pypi/pytz/

Can also be obtained in Anaconda Python.

numpy

numpy-1.6.2.tar.gz

Available for download at: https://pypi.python.org/pypi/numpy/1.6.2

Can also be obtained in Anaconda Python.

matplotlib

matplotlib-1.2.0.tar.gz

Available for download at: https://pypi.python.org/pypi/matplotlib/1.2.0

Can also be obtained in Anaconda Python.

netCDF4 python

netCDF4 1.1.7

Available for download at:https://pypi.python.org/pypi/netCDF4

Can also install using Anaconda Python.

apache tomcat and modwsgi

<https://code.google.com/p/modwsgi/wiki/InstallationOnLinux>

http://tomcat.apache.org

INSTALLATION STEPS

0) Install Python and associated packages listed above

1) Copy the Python scripts and wsgi modules located in

pikalert/scripts/python/backend\_server/ to a directory where

you want them to reside. This directory should be specified in your

PYTHONPATH environment variable to be accessible.

2) Install Apache Tomcat and modwsgi

3) Modify vdt\_apache\_config located in

pikalert/build\_install/vdt\_apache\_config so that the wsgi files

mentioned in 1) can be found.

4) Modify the directories mentioned in

backend\_sys\_path.py. Backend\_sys\_path.py contains all the directory

path information used by the python backend server scripts.

Note that the information for Alaska need not be modified since Alaska

is not currently supported by this particular release of the software.

Here is more detailed guidance for configuring Michigan (analogous for

other states):

Michigan\_district\_alerts\_dir = "your\_choice"

Michigan\_latest\_vehicles\_dir = "should be set to the output location of probe\_to\_json.py for this state"

Michigan\_maw\_json\_dir = "should be set to the output of rwa\_to\_json.py for this state"

Michigan\_obs\_stats\_dir = "should be set to the vdt stats output directory typically called mi\_vdt\_output"

Michigan\_obs\_stats\_json\_dir = "should be set to the obs stats JSON directory, one of the output directories of the rwa"

Michigan\_plots\_dir = "should be set to the plot data JSON directory, one of the output directories of the rwa"

Michigan\_road\_segments\_file = "should be set to the road segment file for this state such as "mi\_roads.20140613.nc"

Michigan\_rwh\_dir = "should be set to the location of the mi\_rwh\_output directory"

Michigan\_vdt\_output\_dir = "should be set the directory location of the vdt output"

Michigan\_radius = 20 # in meters

5) Try running:

python backend\_sys\_path.py

to guarantee that no configuration mistakes have been made

6) Restart the apache server so that it can read your new apache

configuration file mentioned in 3)

7) Check to make sure that your configuration in 3) is working correctly:

curl http://your\_machine\_name:8080/latest\_vehicles\?path=/latest\_vehicles\&state=minnesota

curl http://your\_machine\_name:8080/latest\_vehicles\?path=/latest\_vehicles\&state=michigan

curl http://your\_machine\_name:8080/latest\_vehicles\?path=/latest\_vehicles\&state=nevada

curl http://your\_machine\_name:8080/datatime\?path=/datatime\&state=minnesota

curl http://your\_machine\_name:8080/datatime\?path=/datatime\&state=michigan

curl http://your\_machne\_name:8080/datatime\?path=/datatime\&state=nevada

You should see reasonable output from these.

For example:

curl <http://eskimo:8080/latest_vehicles\?path=/latest_vehicles\&state=minnesota>

should yield something like:

{"data\_time":"201504162315","districts":[{"display\_name":"minnesota","district\_name":"minnesota","max\_lat":49.70000076293945,"max\_lon":-88.5,"min\_lat":43.40000152587891,"min\_lon":-97.69999694824219,"vehicles":[{"auto\_brake":"-9999","heading\_deg":"272.1","id":"209552","lat":"45.2207","lon":"-93.0353","obs\_time":"1429226370","road\_temp\_f":"-9999.0","speed\_mph":"0","temp\_f":"-\

9999.0","traction\_control":"-9999","wiper\_status":"1"}]}]}

The command:

curl <http://eskimo:8080/datatime\?path=/datatime\&state=michigan>

should yield something like:

[{"latest\_time": "201504162135", "dir": "latest\_vehicles"},{"latest\_time": "201504162318", "dir": "rec\_treatment"},{"latest\_time": "201504162315", "dir": "road\_wx\_dir"}, {"latest\_time":"20\

1504162300", "dir": "district\_alerts"}]

## 6 PIKALERT VDT, EMDSS, AND MAW SYSTEMS OVERVIEW

## 6.1 System Goals

The systems described herein are designed to be functional prototypes to address needs of the road weather community. Specifically, the systems are designed to fulfill the following goals:

* Pikalert VDT: Collect, quality check, and aggregate mobile data along with ancillary weather observations. The additional Road RWH module was designed to take output from the VDT and produce hazard assessments of precipitation, road conditions, and visibility along the configured road segments.
* EMDSS: Enhance the current MDSS with output from Pikalert. One of the major benefits of this enhancement is filling in the gaps between RWIS stations with mobile data. The EMDSS provides an assessment and forecast along road segments of road weather conditions as well as treatment recommendations (where/when to plow and/or apply chemical) for maintenance supervisors.
* MAW: Provide road weather assessments and forecasts along configured road segments for the traveling public to aid in decision making processes about when/where/if to travel during adverse weather.

To make it easier to evaluate and/or implement these components, source code is provided within this release for all of the modules. This should enable the extraction of pieces of the system for incorporation within another system. The current Pikalert release can be obtained by registering at the following website:

<http://ral.ucar.edu/projects/connected_vehicles/pikalert/>

The Pikalert system, EMDSS, and MAW ingest weather data at locations important to the users’ operations. These forecast locations are typically placed at surface observation stations such as RWIS and METAR sites as well as Pikalert road segments. For the EMDSS and MAW, the weather forecasts at each forecast location serve as input to the pavement heat balance model (e.g., METRo) that predicts the road surface and subsurface temperatures and the snow depth at each forecast lead time. In EMDSS, these forecast road conditions are used to generate treatment plans at based on Rules of Practice guidelines.

The systems are designed to be modular so that those interested in specific components could utilize those capabilities without needing all the components. For example, if implementers had access to a different pavement temperature model, they could swap out METRo and replace it with their preferred model. Implementers can download, purchase, or otherwise generate weather forecasts to drive the Road Condition Treatment Module (RCTM). The components can be run separately and implementers do not have to run all the prototype components, although various components do require specific inputs and data formats (e.g., in order to display hazards the MAW requires assessed and forecasted hazards as inputs).

The Pikalert, EMDSS, and MAW applications are a research and development effort. As such, overall reliability of the system code has not been exhaustively evaluated.

## 6.2 System Structure

The Pikalert, EMDSS, and MAW applications consist of ingest processes, algorithm processes, and display processes. This section gives a brief overview of the hardware and software architecture of the systems. More detail can be found in the System Design Description referenced in Table 1.

### 6.2.1 Computer Hardware

The computational requirements for the Pikalert VDT, EMDSS, and MAW systems are related to the number of road segments configured in the system, the update cycle, and the amount of vehicle and ancillary data that must be processed. The following recommended hardware specifications are based on the current systems running at NCAR over interstate highways in three different states. The road segments are configured every one mile and updated every five minutes. Additional assessments and forecasts are made at RWIS station locations.

**Recommended Hardware Based On Three State System**

Quad-core 3.00 GHz processor

2 TB disk space

32 GB RAM

### 6.2.2 Software Architecture

The application algorithms and code have been designed to run on common LINUX workstations. Currently these subsystems have only been compiled and tested under the Linux Debian Wheezy operating system. The display has been developed using Sencha Touch (<http://www.sencha.com/products/touch/>). As such, the html5 code is machine independent and should run on any properly configured hardware system.

The applications are part of, and in and of themselves, a modular system. With such a modular system, a user can run only those pieces that are relevant to their application (e.g., the Pikalert VDT to produce road segment statistics). However, systems further downstream such as the MAW require proper inputs to function properly (e.g., outputs like those from the Pikalert VDT).

### 6.2.3 Computer Languages

System components are coded in the following languages:

* C++ (VDT, RWH, RWA, probe\_to\_json)
* Javascript (EMDSS and MAW web displays)
* Java (MAW phone application)
* Python (system glue layer, backend server modules)
* FORTRAN, C++ (METRo)

### 6.2.4 External Software Requirements

A number of third-party software packages are required to run the entire Pikalert system and the EMDSS and MAW applications. These packages are freely available and must be installed before installing the Pikalert, EMDSS, and MAW systems code. Some of the packages may come pre-installed with the OS, or it may be necessary to install them as Debian packages (preferred) or they can be built from scratch. The table below lists these packages, the minimum or specific version required, and where they can be obtained if the OS’s package manager does not include them.

The package versions listed here are the minimum requirement for the system and with which it has been tested. More up-to-date packages may also work but have not been tested. It is recommended to use the OS’s package manager to install these files to ensure dependencies are met.

Sometimes packages include a companion developer (package name-dev) package. If a particular third-party software package has a dev package available as, please install it as well.

Scons 2.3.0 http://www.scons.org/download.php

libconfig 1.4.9 http://www.hyperrealm.com/libconfig/

ANN 1.1.2 http://www.cs.umd.edu/~mount/ANN/

NetCDF4 4.2.1.1 <http://www.unidata.ucar.edu/downloads/>

netcdf/index.jsp

NetCDF4 C++ API 4.2 http://www.unidata.ucar.edu/downloads/

netcdf/netcdf-cxx/index.jsp

HDF5 1.8.10 ftp://ftp.hdfgroup.org/HDF5/current/src/

zlib 1.2.5 http://zlib.net/

Boost 1.51.0 http://sourceforge.net/projects/boost/

files/boost/1.51.0/

Blitz 0.9 http://sourceforge.net/projects/blitz/

files/blitz/Blitz%2B%2B%200.9/

Python-2.7.1 2.7.1 http://www.python.org/download/releases/

2.7.1/ (NOTE: It may be more convenient to install Anaconda Python since it contains most of the third party Python packages required by the system such as numpy, matplotlib, etc.. See: https://store.continuum.io/cshop/anaconda)

pytz 2013.9-py2.7 https://pypi.python.org/pypi/pytz/

numpy 1.6.2 https://pypi.python.org/pypi/numpy/1.6.2

matplotlib 1.2.0 https://pypi.python.org/pypi/matplotlib/

1.2.0

json\_spirit 4.06 http://www.codeproject.com/Articles/

20027/JSON-Spirit-A-C-JSON-Parser-

Generator-Implemented

udunits 2.1.24 https://www.unidata.ucar.edu/downloads/

udunits/index.jsp

ldm <http://www.unidata.ucar.edu/software/ldm/>

Scons is used to build the VDT executables rather than GNU ‘make’. The tool finds the appropriate installed compilers. If the installation has issues with compilers, it is recommended to have installed the GNU compiler set (gcc, g++, and gfortran).

### 6.2.5 Data Ingest

Several types of live data[[2]](#footnote-2) are used by the system. These data include numerical weather forecasts, weather and road observations, and gridded data sets such as radar and satellite data. Some of these data sets are made available to the display after simple re-formatting, but many are used by the system processes in creating road forecasts and alerts. The downloading, reformatting, or creation of the raw data sets is beyond the scope of this document. It is assumed that these data sets are on disk and available in the appropriate formats for use by the system. Note that setting up real-time data for an operational system can be a major undertaking and may require significant knowledge of meteorological and other data sets.

## 6.3 Application Components Overview

The Pikalert, EMDSS, and MAW applications consist of individual components which have well-defined roles in the larger system. This section presents a high level description of the components that go into making the three systems. Figure 6.1 provides an overview of the Pikalert data flow.

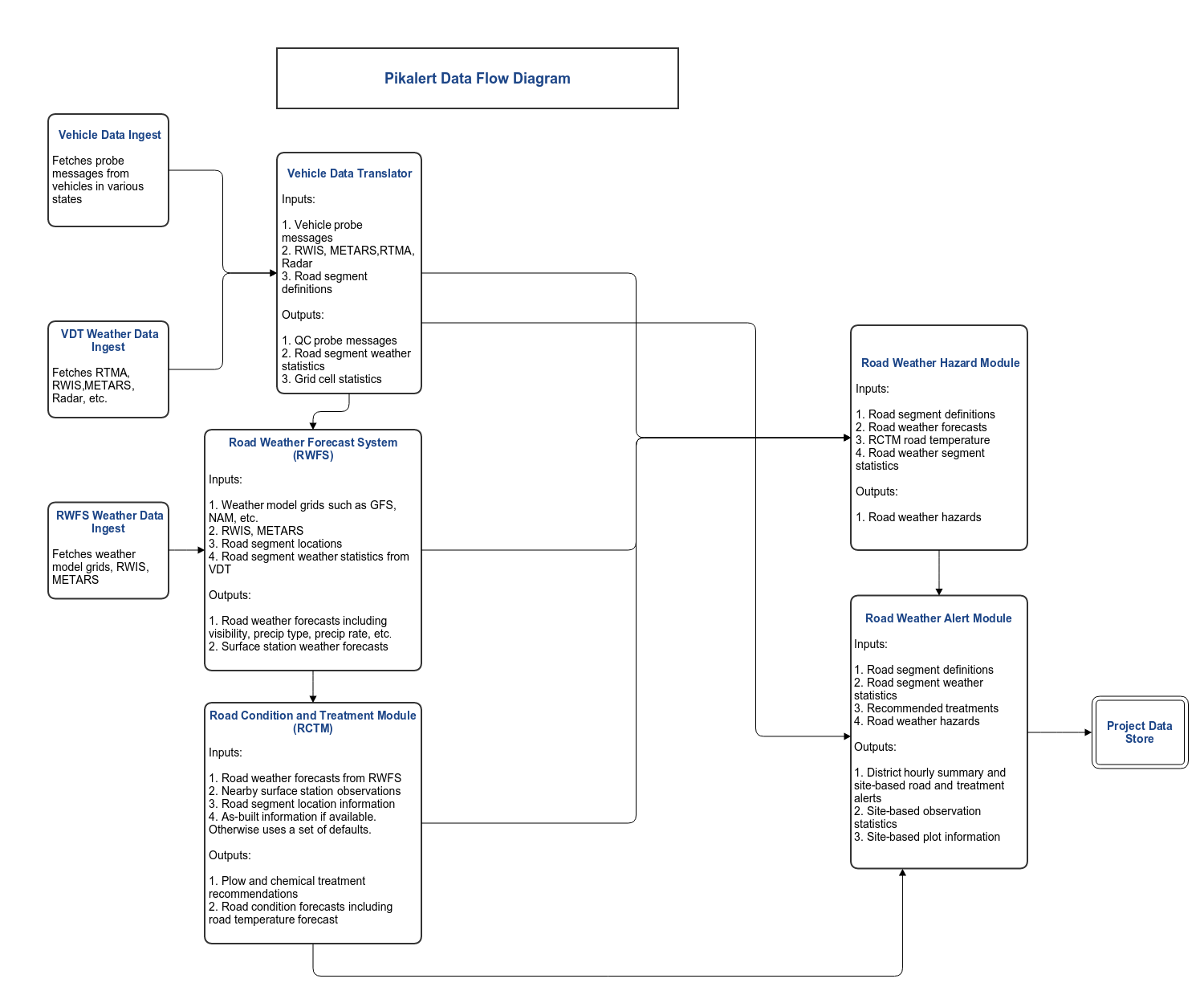


Figure 6.1 Overview of Pikalert, EMDSS, and MAW systems.

### 6.3.1 Weather Data Ingest

The Pikalert VDT currently ingests weather data as presented in Figure 6.2.

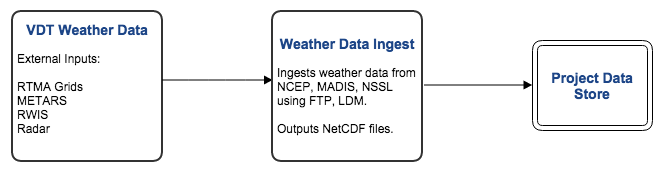


Figure 6.2 Pikalert VDT Weather Ingest Steps

### 6.3.2 Vehicle Data Translator

Figure 6.3 gives an overview of the Pikalert VDT process.

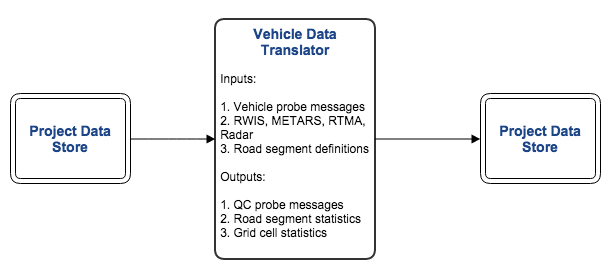


Figure 6.3 Pikalert VDT Process Overview

### 6.3.3 Road Weather Forecast System Diagram

The RWFS process is described in Figure 6.4.

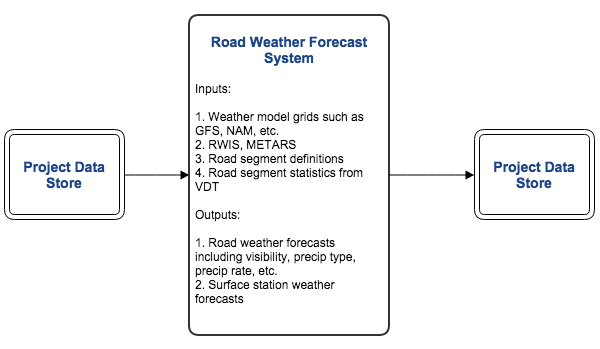


Figure 6.4 Overview of the RWFS Process

### 6.3.4 Road Condition and Treatment Module

The RCTM process is described in Figure 6.5.

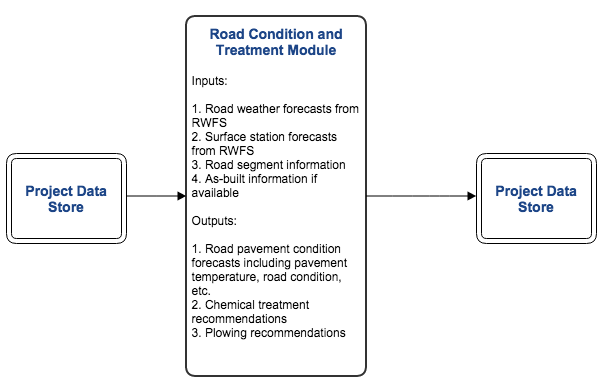


Figure 6.5 RCTM Process Overview

### 6.3.5 Road Weather Hazard Module

The RWH process is described in Figure 6.6.

# 

Figure 6.6 Overview of the RWH Process

### 6.3.6 Road Weather Alert Module

Figure 6.7 describes the RWA process.



Figure 6.7 RWA Processing Steps

### 6.3.7 EMDSS Module

Figure 6.8 describes the EMDSS module consisting of the EMDSS display and the back end server supporting the display.

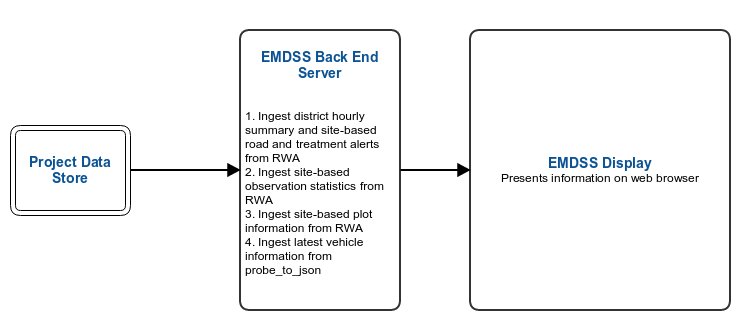


Figure 6.8 EMDSS Processing Steps

### 6.3.8 MAW and MAW Phone Application Modules

Figure 6.9 describes the MAW and MAW phone application processes.

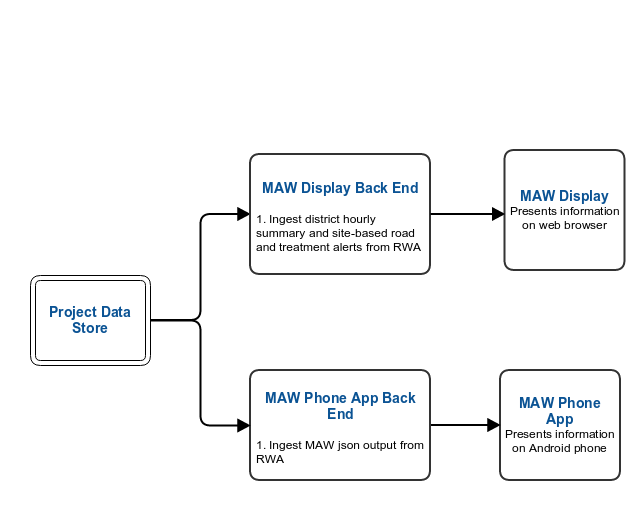


Figure 6.9 MAW Processing Steps

# 7 SYSTEM OPERATIONS

Python scripts prefixed with ep\_run\_ were written to control the Pikalert system processing for the different states. These can scheduled by establishing a suitable crontab in the system login account. The Sencha Touch documentation (Section 6.2.2) provides information on running the display capabilities of the EMDSS and MAW applications.

The EMDSS contains software known as the system glue layer. The glue layer contains knowledge of the file system layout for EMDSS. Information about this can be found in the MDSS technical guide ([http://ral.ucar.edu/projects/rdwx\_mdss/documents /MDSS\_Tech\_Description\_15Nov2007.pdf](http://ral.ucar.edu/projects/rdwx_mdss/documents%20/MDSS_Tech_Description_15Nov2007.pdf)).

Additional information about the system design of the Pikalert VDT, EMDSS, and MAW can be found in the System Design Document. See Table 1, the Related Documents table.

# 8 CONFIGURATION FILES

## 8.1 Configuration File Overview

The Pikalert, EMDSS, and MAW applications use configuration files in various formats to tailor the ultimate output of the system. The configuration files are plain-text ASCII files aside from the road segment file and are described below.

## 8.2 Road Segment Files

In order to run the Pikalert VDT, road segments must be defined for the system. This information is placed in a netCDF file containing the following variables:

variables:

double latitude(point\_num) ;

latitude:long\_name = "latitude" ;

latitude:standard\_name = "latitude" ;

latitude:valid\_range = -90., 90. ;

latitude:units = "degrees\_north" ;

double mid\_point\_latitude(point\_num) ;

mid\_point\_latitude:long\_name = "latitude of the road segment mid point" ;

mid\_point\_latitude:standard\_name = "mid point latitude" ;

mid\_point\_latitude:valid\_range = -90., 90. ;

mid\_point\_latitude:units = "degrees\_north" ;

double longitude(point\_num) ;

longitude:long\_name = "longitude" ;

longitude:standard\_name = "longitude" ;

longitude:valid\_range = -180., 180. ;

longitude:units = "degrees\_east" ;

double mid\_point\_longitude(point\_num) ;

mid\_point\_longitude:long\_name = "longitude of the road segment mid point" ;

mid\_point\_longitude:standard\_name = "mid point longitude" ;

mid\_point\_longitude:valid\_range = -180., 180. ;

mid\_point\_longitude:units = "degrees\_east" ;

float elevation(point\_num) ;

elevation:long\_name = "elevation" ;

elevation:standard\_name = "elevation" ;

elevation:units = "feet" ;

elevation:\_FillValue = -9999.f ;

char seg\_name(point\_num, seg\_name\_len) ;

seg\_name:long\_name = "road identifier number for point" ;

seg\_name:standard\_name = "road identifier" ;

seg\_name:units = "" ;

int seg\_id(point\_num) ;

seg\_id:long\_name = "segment identifier for point" ;

seg\_id:standard\_name = "segment identifier" ;

char aux\_id(point\_num, aux\_id\_len) ;

aux\_id:long\_name = "auxiliary identifier" ;

aux\_id:standard\_name = "auxiliary identifier" ;

int point\_type(point\_num) ;

point\_type:long\_name = "point type" ;

point\_type:flag\_values = 0, 1 ;

point\_type:flag\_meanings = "road\_seg rwis\_site" ;

char rwfs\_id(point\_num, rwfs\_id\_len) ;

rwfs\_id:long\_name = "corresponding rwfs site id" ;

rwfs\_id:standard\_name = "road identifier" ;

char road\_type(point\_num, road\_type\_len) ;

road\_type:long\_name = "road type" ;

short route\_class(point\_num) ;

route\_class:long\_name = "class route" ;

char highway\_type(point\_num, highway\_type\_len) ;

highway\_type:long\_name = "highway type" ;

float speed\_mph(point\_num) ;

speed\_mph:long\_name = "speed miles per hour" ;

speed\_mph:standard\_name = "speed\_mph" ;

speed\_mph:valid\_range = 0., 120. ;

speed\_mph:units = "mi" ;

The road segment files for Nevada, Minnesota and Michigan are in the Pikalert distribution in the following locations:

pikalert/static\_data/cdl/nv\_roads.20131111.nc

pikalert/static\_data/cdl/mn\_roads.20131111.nc

pikalert/static\_data/cdl/mi\_roads.20140613.nc

## 8.3 VDT Configuration File

The VDT configuration file contains the following information:

* RTMA projection information
* A lat/lon box around the area of interest
* A link to a climatology netCDF file and other config input files
* Probe message field names
* QCh test configuration information and bounds for the tests

There is a VDT configuration file for Minnesota, Michigan and Nevada in the Pikalert distribution. Here are their locations:

pikalert/static\_data/config/vdt\_mn\_config\_new.cfg

pikalert/static\_data/config/vdt\_mi\_config\_new.cfg

pikalert/static\_data/config/vdt\_nv\_config\_new.cfg

Information on how to modify these configuration files is contained in the above files.

Note that the configuration file format is specified by the libconfig library located at:

<http://www.hyperrealm.com/libconfig/libconfig_manual.html>.

## 8.4 RWH Configuration File

This file contains information on the fields that are read into the RWH. It should not be edited due to the importance of reading in each variable to the scientific algorithms in the RWH. Adding variables to the file will not improve the RWH system because the source code must be edited in order for the algorithms to use new variables.

Here is the location of the RWH configuration in the Pikalert distribution:

pikalert/static\_data/config/rwh\_config\_new.cfg

Note that the configuration file format is specified by the libconfig library located at:

<http://www.hyperrealm.com/libconfig/libconfig_manual.html>.

## 8.5 RWA Configuration File

The RWA configuration file controls the types of assessments and forecasts from RWH that warrant advisories and warnings. Each condition consists of one or more “EQ” tuples that identify the condition. The last tuple contains alert information consisting of: alert class, alert type, alert severity (missing, clear, warning, alert), alert time class (obs, fcst, both), phone app conditions, phone app actions, phone app priority, phone app condition id, and phone app alert action code.

An example of an RWA configuration file entry is:

(("EQ", "pavement\_condition", 3), ("EQ", "pavement\_slickness", 1), ("pavement", "slick, icy", "alert", "both", "icy, slick roads", "delay travel, seek alternate route, or drive slowly and use extreme caution", 23, 5, 3))

The first tuple (“EQ”, “pavement\_condition”, 3) indicates that the RWH output for pavement\_condition must be equal to 3, which is icy pavement. Additionally, (“EQ”, “pavement\_slickness”, 1) indicates the RWH output for pavement\_slickness must also be 1, indicating slippery roads. The following tuple states that:

* This condition is alert class “pavement”
* The alert type is “slick, icy”
* Severity is “alert”
* This alert type applies to “both” forecast and observed conditions of slick roads and icy pavement
* The phone app will list conditions as “icy, slick roads”
* The phone app will recommend the following action: “delay travel, seek alternate route, or drive slowly and use extreme caution”
* The phone app priority, condition id, and alert action code is 23, 5, and 3 respectively

There is an RWA configuration file for Minnesota, Michigan and Nevada in the Pikalert distribution. Here are their locations:

pikalert/static\_data/config/ mn\_rwa.cfg

pikalert/static\_data/config/mi\_rwa.cfg

pikalert/static\_data/config/nv\_rwa.cfg

Information on how to modify these configuration files is contained in the above files.

Note that the configuration file format is specified by the libconfig library located at:

<http://www.hyperrealm.com/libconfig/libconfig_manual.html>.

## 8.6 Site List Configuration Files

Site list files are used by various components of the system in order to process and output data for specific sites, particularly the forecast engine and RCTM. The site list files are text files that contain information about known sites, one site per line. The fields in each line are separated by a semi-colon. Each line has the following format:

SITE\_ID;WMO\_ID;ICAO\_ID;LAT;LON;ELEV;REGION;NAME;STATE;COUNTRY

where:

SITE\_ID = a unique 8-digit number

WMO\_ID = WMO ID number of site (-9999 if not a WMO site)

ICAO\_ID = ICAO or other abbreviation for site ("----" if not known)

LAT = latitude in decimal degrees (negative for south latitude)

LON = longitude in decimal degrees (negative for west longitude)

ELEV = elevation above sea level in meters

REGION = global region defined by WMO (North America == 4)

NAME = the display name of the site

STATE = two-letter state abbreviation

COUNTRY = country name (e.g. UNITED STATES)

Note that site list files should be sorted in ascending order based on the SITE\_ID, each of which must be unique. Site list files are parsed using the ";" delimiter, so there is some leeway in the size of certain fields. However it is best to make any added sites conform to existing sites as much as possible.

## 8.7 Back end server configuration files

The back end server uses the following configuration files:

backend\_sys\_path.py – contains all the directory path information used by the Python backend server scripts

phone\_ids.py – contains the registry of phone numbers that will be allowed to use the MAW phone application

Both of these configuration files are located in:

pikalert/scripts/python/backend\_server/wsgi

Information on how to modify these configuration files is contained in the above files.

# 9 DATA INGEST

No data ingest infrastructure is specified for the system. Several data sets are required by the applications. These can be generated manually, purchased, or downloaded from the Internet. This document does not specify or suggest how the input data are generated.

1. Pikalert® is a registered trademark of the University Corporation for Atmospheric Research (UCAR) [↑](#footnote-ref-1)
2. Live data refers to data that flow into the system shortly after they are measured or obtained. [↑](#footnote-ref-2)