The Maintenance Decision Support System (MDSS) Project

MDSS Prototype
Release-3.0
Technical Description

Version 1.0

19 November 2004

Prepared for:
Federal Highway Administration (FHWA)
Road Weather Management Program

Prepared by:
MDSS National Laboratory Consortium
(NCAR, LL, FSL, CRREL)
RELEASE NOTES

<table>
<thead>
<tr>
<th>Version Number</th>
<th>Date</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1.0</td>
<td>19 November 2004</td>
<td>Revised based on a fall 2004 FHWA review.</td>
</tr>
</tbody>
</table>

Questions and comments about this document should be directed to:

Dr. William Myers
Research Applications Laboratory
NCAR
P.O. Box 3000
Boulder, CO 80307
Ph: 303-497-8412
E-mail: myers@ucar.edu

Jim Cowie
Research Applications Laboratory
NCAR
P.O. Box 3000
Boulder, CO 80307
Ph: 303-497-2831
E-mail: cowie@ucar.edu
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION</td>
<td>8</td>
</tr>
<tr>
<td>2. INTENDED AUDIENCE</td>
<td>8</td>
</tr>
<tr>
<td>3. BACKGROUND</td>
<td>8</td>
</tr>
<tr>
<td>3.1 Disclaimer</td>
<td>9</td>
</tr>
<tr>
<td>3.2 Overview of System Improvements since Release-2</td>
<td>9</td>
</tr>
<tr>
<td>4. RELATED DOCUMENTS</td>
<td>11</td>
</tr>
<tr>
<td>5. SYSTEM OVERVIEW</td>
<td>12</td>
</tr>
<tr>
<td>5.1 System Goals</td>
<td>12</td>
</tr>
<tr>
<td>5.2 System Structure</td>
<td>13</td>
</tr>
<tr>
<td>6. SYSTEM OPERATIONS</td>
<td>25</td>
</tr>
<tr>
<td>6.1 Process Invocation</td>
<td>25</td>
</tr>
<tr>
<td>6.2 Display Invocation</td>
<td>26</td>
</tr>
<tr>
<td>7. PROCESS ERROR HANDLING</td>
<td>26</td>
</tr>
<tr>
<td>7.1 Data Logging</td>
<td>26</td>
</tr>
<tr>
<td>7.2 Error Handling</td>
<td>28</td>
</tr>
<tr>
<td>8. INTERPROCESS COMMUNICATION</td>
<td>29</td>
</tr>
<tr>
<td>8.1 File System Communication</td>
<td>29</td>
</tr>
<tr>
<td>8.2 File Formats</td>
<td>30</td>
</tr>
<tr>
<td>8.3 Display Request Interprocess Communication</td>
<td>31</td>
</tr>
<tr>
<td>9. Configuration Files</td>
<td>31</td>
</tr>
<tr>
<td>9.1 Configuration File Overview</td>
<td>31</td>
</tr>
<tr>
<td>9.2 Site List Configuration Files</td>
<td>31</td>
</tr>
<tr>
<td>10. REDUNDANCY</td>
<td>32</td>
</tr>
<tr>
<td>11. DATA INGEST</td>
<td>33</td>
</tr>
<tr>
<td>11.1 Identification</td>
<td>33</td>
</tr>
<tr>
<td>11.2 Type</td>
<td>33</td>
</tr>
<tr>
<td>11.3 Purpose</td>
<td>33</td>
</tr>
<tr>
<td>11.4 Function</td>
<td>33</td>
</tr>
<tr>
<td>11.5 Dependencies</td>
<td>33</td>
</tr>
<tr>
<td>11.6 Interface</td>
<td>34</td>
</tr>
<tr>
<td>11.7 Processing</td>
<td>35</td>
</tr>
<tr>
<td>12. ROAD WEATHER FORECAST SYSTEM (RWFS)</td>
<td>36</td>
</tr>
<tr>
<td>12.1 Identification</td>
<td>36</td>
</tr>
<tr>
<td>12.2 Type</td>
<td>36</td>
</tr>
<tr>
<td>12.3 Purpose</td>
<td>36</td>
</tr>
<tr>
<td>12.4 Dependencies</td>
<td>36</td>
</tr>
<tr>
<td>12.5 Interfaces</td>
<td>37</td>
</tr>
<tr>
<td>12.6 Processing</td>
<td>37</td>
</tr>
<tr>
<td>13. ROAD CONDITION AND TREATMENT MODULE (RCTM)</td>
<td>39</td>
</tr>
<tr>
<td>13.1 Identification</td>
<td>39</td>
</tr>
<tr>
<td>13.2 Type</td>
<td>39</td>
</tr>
<tr>
<td>13.3 Purpose</td>
<td>39</td>
</tr>
<tr>
<td>13.4 Function</td>
<td>40</td>
</tr>
<tr>
<td>13.5 Dependencies</td>
<td>40</td>
</tr>
</tbody>
</table>
APPENDICES

Appendix A: Road Weather Forecast System......................................................78
Appendix B: Road Temperature Model (SNTHERM-RT)......................................86
Appendix C: Chemical Concentration Algorithms..............................................89
Appendix D: Net Mobility Module.......................................................................101
Appendix E: Rules of Practice Module.................................................................102
Appendix F: Appendix F: Characterize Storm Module........................................112
Appendix G: Precipitation Type Algorithms.........................................................119
Appendix H: Experimental Blowing Snow Algorithm..........................................121
Appendix I: Installing the MDSS from CD-ROM................................................123
Appendix J: Technical Points of Contact.............................................................124
Acronym Glossary

AASHTO – American Association of State Highway and Transportation Officials
AVN – National Weather Service Aviation model (now known as the GFS)
CDL – Command Description Language
CRREL – U.S. Army Cold Regions Research and Engineering Laboratory
DICAST™ – Dynamic Integrated foreCAST System
DMOS – Dynamic Model Output Statistics
DOT - Department of Transportation
DSS – Decision Support System
EMFP - Ensemble Model Forecast Provider
ESRI – Environmental Systems Research Institute
Eta – National Weather Service Model
ETL – NOAA, Environmental Technology Laboratory
FAA – Federal Aviation Administration
FHWA – Federal Highway Administration
FM – Forecast Module
FP – MDSS Functional Prototype
FSL – NOAA, Forecast Systems Laboratory
FTP – File Transfer Protocol
GFS – National Weather Service Global Forecasting System model (formerly AVN)
GRIB – GRIdded Binary
HOTO - Office of Transportation Operations
IADOT - Iowa Department of Transportation
JDK – Java Development Kit
LDADS – Local Data Acquisition and Dissemination System
LDM – Local Data Manager
MADIS – Meteorological Assimilation Data Ingest System (NOAA/FSL)
MAVMOS – Model Output Statistics for the NWS AVN model
MDSS - Maintenance Decision Support System
METAR – Meteorological Surface Observation
MIT/LL - Massachusetts Institute of Technology - Lincoln Laboratory
MM5 – Mesoscale Model – Version 5 (NCAR & Penn State)
MOS – Model Output Statistics
NCEP - National Centers for Environmental Prediction
NetCDF – Network Common Data Format
NSF – National Science Foundation
NSSL – NOAA, National Severe Storms Laboratory
NOAA – National Oceanic and Atmospheric Administration
NCAR - National Center for Atmospheric Research
NVD - Non-Verifiable Data
NWS – National Weather Service
OCD – Operational Concepts Description
RAMS – Regional Atmospheric Modeling System (Colorado State University)
RAL – Research Applications Laboratory, NCAR (formerly RAP)
RAP – Research Applications Program, NCAR
RCTM – Road Condition & Treatment Module
RWIS – Road Weather Information System
RWFS – Road Weather Forecast System
RWMP - Road Weather Management Program
STWDSR - Surface Transportation Weather Decision Support Requirements
SNTHERM – Pavement temperature model provided by CRREL
TUNL – Treatment Update Network Layer
UCARF – University Corporation for Atmospheric Research Foundation
UNIDATA – University Data program of the UCAR Office of Programs
WIST-DSS - Weather Information for Surface Transportation Decision Support System
WMO – World Meteorological Organization
WRF – Weather Research & Forecasting Model
1 INTRODUCTION

This document describes Release 3.0 of the prototype Maintenance Decision Support System (MDSS) technical components (e.g., code and processes) and is organized into two main parts: a main body and a set of appendices. The main body contains technical descriptions of the prototype MDSS software, whereas the appendices include descriptions of algorithms and techniques used in the system. The format of the main body is based on the Institute of Electrical and Electronics Engineers (IEEE) standard for Software User Documentation (ANSI/IEEE standard 1063-1987). This document is organized so that high-level system descriptions are presented first followed by detailed descriptions of each system component. The detailed descriptions are presented as software processes covering process type, purpose, function, dependencies, and interfaces.

2 INTENDED AUDIENCE

The intended audience of this document is software engineers with extensive knowledge of the C++ programming languages and the UNIX operating system. Technical points of contact for the prototype MDSS system are provided in Appendix J.

3 BACKGROUND

This MDSS Project is part of a federal procurement for research projects and deployment advocacy, which is funded through the Intelligent Transportation System (ITS) Joint Project Office (JPO) of the FHWA.

It is envisioned that components of the prototype MDSS system developed by this project will be further developed, integrated with other operational components, and deployed by road operating agencies, including state departments of transportation (DOTs), and generally supplied by the private sector.

Five national research centers have participated in the development of the MDSS. The participating national labs include:

- Army Cold Regions Research and Engineering Laboratory (CRREL)
- National Center for Atmospheric Research (NCAR)
- Massachusetts Institute of Technology - Lincoln Laboratory (MIT/LL)
- NOAA National Severe Storms Laboratory (NSSL)
- NOAA Forecast Systems Laboratory (FSL)
3.1 Disclaimer

The MDSS and its software are prototype systems. The development strategy was to identify advanced scientific and engineering technologies that, if reapplied, could be used to assess the feasibility of an MDSS system. For this reason, the MDSS software should be used with caution. It is not intended to be a fully integrated ‘plug-and-play’ technology. It is anticipated that the prototype MDSS software would be used as a springboard toward the development of commercial road weather systems that contain MDSS 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.

Because the MDSS is only a limited prototype system, not all capabilities desired by DOTs have been incorporated. For example, most DOTs have indicated a desire that an operational MDSS interface with their central database to obtain actual treatment data and that the MDSS output its recommended treatments to a DOT archive. Because these interfaces are specific to each DOT they have not been incorporated into the prototype which is mainly designed to be a generic (i.e., not state specific) proof-of-concept system.

3.2 Overview of System Improvements since Release-2

The MDSS has been refined significantly since Release-2 which was developed for the first Iowa field demonstration during the winter of 2003. Several parts of the MDSS were upgraded based on experience and user feedback from that first season. The Release-3 version was demonstrated during the second Iowa field demonstration which was conducted from December 2003 to April 2004. Personnel in three Iowa DOT maintenance garages used this software during their winter road maintenance operations.

There was a large focus to improve the weather forecast component of the system. In particular changes were made to improve precipitation and short-term forecasts. To achieve these goals, the ensemble modeling system and the RWFS were redesigned to take advantage of higher temporal resolution (hourly) model data. In 2003 all model data coming into the RWFS had 3 hour temporal resolution and were interpolated to hourly within the RWFS. In 2004 the 0-12 hour forecasts were generated from the higher resolution mesoscale model output and these were generated separately within the RWFS from the 12-48 hour forecasts which were mainly based on coarser (3 hourly output) NCEP model data. To improve the precipitation forecasts, development of weight generation algorithms was required to handle special cases of incomplete or poor precipitation observations. Also, a “Forward Error Correction” technique was developed to nudge the near term forecasts close to the observed conditions.

The RCTM modifications included support for bridge temperature forecasts and much improved Rules of Practice code. The method used to initialize the road temperature model (SNTHERM-RT) was changed in 2004 to utilize actual surface and subsurface data from
RWIS rather than using predicted profiles as was the case in 2003. In addition, insolation data from the weather prediction models was used directly in the road temperature calculations rather than inferring the fluxes from cloud cover as was done in the past. The Rules of Practice module was rewritten to include a sub-module for characterizing the entire storm (duration, precipitation type, road temperatures, etc). This meta information was used to drive the type of treatments needed (pre-treatment, chemical vs. plow-only, frequency, etc). In addition, chemical treatments were calculated by determining the amount of chemicals needed to protect the road from freezing based on the chemicals' eutectic properties, the hourly forecasted precipitation type, amount and road temperatures, and the expected chemical dilution (splatter, traffic, runoff, etc).

The display application was upgraded to better present the data to the user. Digital observations and forecast values were added on the main graphics page as an alternative to the colored dots and mouse-over graphics. A new blowing snow product, a placeholder for a road frost product and an event summary graphic that included information on precipitation type probabilities were also added to the display.

A summary of MDSS prototype enhancements since Release-2 is provided below:

Road Weather Forecast System:
- A time-lagged FSL ensemble model configuration was implemented
  - The MM5 and WRF models were “hot started” using LAPS for initialization and the Eta model for boundary conditions
  - Both WRF and Eta models were run hourly
- RWFS was broken into two forecast subsystems
  - Near-Term System with hourly core forecast resolution out to 12 hours
  - Short-Term System with 3 hourly (interpolated to hourly) resolution for 12-48 hour forecasts
- Weighting schemes were developed to modify system learning due to difficulties presented by poor or missing observations.
  - Weights from nearest observing neighbors were used at non-observing sites.
  - Weights for precipitation variables were hard-wired based on expert opinion.
- Thresholds used in determining precipitation occurrence were refined to capture lighter events.
- A blowing snow potential algorithm was developed and implemented.

Road Condition and Treatment Module:
- A heat balance model (based on SNThERM) for bridges was developed and installed.
- RWIS surface and subsurface observations were used to initialize SNThERM-RT at selected sites.
- Insolation data from the weather models were used in the road temperature model calculations.
- A storm characterization module was added to categorize pre-, in- and post-storm weather and road conditions to customize treatment calculations.
Treatment recommendations are now driven by dynamically calculating chemicals needed based on expected chemical dilution and hourly forecasted weather and road conditions. This replaced the method of utilizing more simplistic FHWA table-based guidelines and generic storm characterizations.

RCTM now tracks the expected amount of “available water” on the road surface enabling the system to protect the water from refreezing post-storm.

The Rules of Practice algorithm evolved to better handle diverse weather/road conditions, including:
- Start up with snow on road
- Pre-treatment with liquid brine, as appropriate
- Distinct freezing rain thresholds

Display Application
- Major internal changes were made to support client-server communication.
- Developed a Permissions Manager.
- Developed more complete graphic and tabular data displays.
  - Added ability to view RWIS data on the main screen.
  - Added auto-scaling for time series plots.
  - Added ability to view data values for all parameters at forecast points.
- Added auto-update capability based on live data.
- Added blowing snow potential product.
- Added placeholder for road frost product.
- Added ability to view treatment history (previous 6-hrs).
  - Created both tabular and graphical history views.
- Added “Event Summary” page with graphical probabilistic data products and an overview of the storm and recommended treatments.
- Added ability to reset route initial conditions to no snow and no residual chemical.

4 RELATED DOCUMENTS

For additional information on the MDSS Project, the reader is directed to related project documents listed in Table 1.

Table 1. Related Documents

<table>
<thead>
<tr>
<th>Document and/or Web Sites</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>STWDSR– Operational Concept Description (OCD) Version 2.0</td>
<td>Federal Highway Administration</td>
</tr>
</tbody>
</table>
5 SYSTEM OVERVIEW

5.1 System Goals

The MDSS described herein is designed to be a functional prototype of a decision support tool used by winter road maintenance managers. The end-user products are designed to satisfy requirements laid out in the Surface Transportation Weather Decision Support Requirements (STWDSR) document. In general, the system is designed to provide timely weather and road condition forecasts coupled with road treatment planning tools.

In addition to the primary objective of providing important safety and efficiency related information, the MDSS has been designed to meet a number of secondary goals including flexibility, simplicity, reliability, and accuracy. Details describing how the MDSS is designed to address these goals are given throughout this document.

Not every system integrator will want to use the entire MDSS as is. For this reason, source code is provided along with this release for several of the modules. This should enable the extraction of pieces of the system for incorporation within another system. Source code can be obtained by registering at the MDSS web site:

http://www.rap.ucar.edu/projects/rdwx_mdss/index.html

The MDSS integrates data from various numerical weather prediction models, surface observation information and climatology to produce weather forecasts at a number of forecast points. These forecast points are typically at surface observation stations such as RWIS and METAR sites, though they need not be. The weather forecasts at each forecast location serve as input to a land-surface model that predicts the road surface and subsurface temperature profiles and the snow depth at each forecast lead-time. These forecast road conditions are used to generate treatment plans at each site based on Rules of Practice guidelines. The MDSS provides a graphical user interface designed for easy interpretation by road maintenance managers. This display is also designed to allow the
maintenance manager to generate “what-if” scenarios by setting up customized treatment plans and seeing the resulting predicted road conditions.

The MDSS is a research and development effort. As such, overall reliability of the system code has not been thoroughly evaluated; in particular the Road Condition and Treatment Module (RCTM).

The described system represents the MDSS prototype as developed through April 2004. This includes inter-process communication between the MDSS Functional Prototype display and upstream modules.

For 2004, the MDSS was built and run on the Red Hat Linux 7.3 operating system. The system software is modular and is written in the C++ and FORTRAN programming languages. Porting the system to other UNIX systems is not expected to be difficult.

The display has been designed to run on a variety of platforms. It has been developed as a Java application allowing the display to run on most any platform that supports Java. This allows a variety of end users to use the display on their existing platforms.

The MDSS includes a number of configuration files. To configure this software for a specific region of interest, the installation guide included with this document should be consulted. The configuration of the Road Weather Forecast System based on DICAST™ technology is fairly straightforward. The configuration of the RCTM requires the creation of files describing the subsurface structure and the traffic levels, etc., at each road forecast site - a more complex task. The configuration of the Treatment Update Network Layer (TUNL) requires defining which hosts and users can access components of the system. Finally, the MDSS display configuration will have to be modified for use at another region. All these modifications are described either herein or in the RWFS Installation Guide, which can be found in the /docs directory of the RWFS software CD.

5.2 System Structure

The MDSS consists of ingest processes, algorithm processes, and a display process. This section discusses the hardware and software architecture and the communications and network interfaces of the prototype MDSS.

5.2.1 Computer Hardware

The computational requirements of the MDSS are linearly related to the number of forecast sites. For all MDSS site list configurations seen thus far (~500 forecast sites), one machine is sufficient to run the entire data ingest and algorithms subsystems. The MDSS is scalable and if the number of forecast sites is sufficiently large, more machines with the same configuration will be required or a single machine with faster processors and more memory will be required.
The display can also be run on this same single platform, but it is envisioned that users at remote sites would most likely be running a display locally. The system described here assumes that the data ingest and algorithms run on a single machine and the output is communicated to the end-users machines via the Internet.

Since processes may be user-request driven, the system load will be slightly more unscheduled and unpredictable than described above. These user requests will generally be rare and require a relatively small amount of processing. However, if the user base is large enough and enough requests are generated, the computational resources of the system may be taxed. In this unlikely case, it may be necessary to add further computational resources.

The MDSS hardware specification for FY 2004 is displayed below. Requirements for running only the display PC component of the system are less than what is required for running the RWFS and RCTM components. Minimal PC requirements are shown. Additional memory and a faster processor should provide better display performance.

**RWFS and RCTM components**

- Dual 3.0 GHz processors
- 250 GB disk space
- 2.0 GB memory

**Display PC**

- 166 MHz Pentium processor (400 MHz or faster is desired)
- 500 MB disk space
- 256 MB memory

The cost of each of these configurations is less than $5,000.

Note: The hardware listed above does not include any hardware necessary to run the supplemental numerical weather forecast models that are being provided by FSL for the MDSS prototyping effort. For information on the hardware requirements for the FSL models, please see the file titled “docs\overview.doc”, which is located on the Release-3 FSL Tailored Numerical Weather Forecasting disk.

### 5.2.2 Communications and Network Interfaces

The MDSS uses standard TCP/IP protocols. This is the underlying protocol standard for most communication on the Internet. An Internet link with at least T1 data rates is required. This link allows timely download of large numerical weather prediction data sets, but is not required for the display component of the system.
If input data redundancy is desired (see Section 10 of this document), a NOAA Port satellite downlink system may be purchased. In this case, the dish vendor should be consulted for interface considerations.

If more than one machine is required to run the MDSS system, the data disks should be cross-mounted (NFS) on a Local Area Network (LAN). In this way, all data will appear to be local on any machine. I/O is generally not a bottleneck for the internal processing of the data.

### 5.2.3 Software Architecture

The MDSS data ingest and algorithms have been designed to run on common UNIX workstations. Currently these subsystems have only been compiled and tested thoroughly under the Intel-based Linux Red Hat operating system. The display has been developed as a Java application. As such, the binary-generated Java byte code is machine independent and should run on any properly configured hardware system as specified in section 16.5.1.

The MDSS is a distributed system. It consists of a number of independent processes each performing a straightforward, well-defined function. The individual modules of the MDSS have been designed to be relatively simple. Each process, while perhaps doing sophisticated processing, has been designed to know very little about the outside world. For example, the processes know nothing about the file system, nor do they consult the system clock to find out the current time. Instead, all information required for processing, such as file names and relevant time parameters are passed to the process as command line arguments. The rationale behind this design is to create a system that is both reliable and maintainable.

One major advantage of this methodology is that every instance of every process is completely repeatable. Log files described later keep track of the command line executed as well as the status of each process run. Bugs can easily be traced by repeatedly running the code with the same command line.

### 5.2.4 Computer Languages

Each component of the system is coded in one of the following languages:

- C/C++
- Java
- FORTRAN
- Perl
- Python
Most of the MDSS algorithm code is written in C++. The exceptions for compiled code are the road temperature and snow depth model, SNTHERM-RT, and related modules used in the IR flux calculations which are written in FORTRAN.

The scripting languages, Perl and Python, are used for data reformatting, server-side request processing (CGI handling of display requests), and process invocation scripts. To the greatest extent possible, C and C++ code is POSIX compliant.

5.2.5 External Software Requirements

A number of third-party software packages are required to run the entire prototype MDSS system. These packages are freely available and must be installed before installing the MDSS system code. The table below lists these packages, the minimum version required and where they can be obtained. The exception is the GNU gcc compiler which requires the exact version listed.

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
<th>Website</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNU gcc</td>
<td>2.9.5</td>
<td><a href="http://gnu.org/">http://gnu.org/</a></td>
</tr>
<tr>
<td>Perl</td>
<td>5.6</td>
<td><a href="http://www.cpan.org/">http://www.cpan.org/</a></td>
</tr>
<tr>
<td>Unidata LDM</td>
<td>6.0.14</td>
<td><a href="http://www.unidata.ucar.edu/packages/ldm">http://www.unidata.ucar.edu/packages/ldm</a></td>
</tr>
<tr>
<td>Unidata netCDF</td>
<td>3.5</td>
<td><a href="http://www.unidata.ucar.edu/packages/netcdf">http://www.unidata.ucar.edu/packages/netcdf</a></td>
</tr>
<tr>
<td>Unidata netCDF-perl</td>
<td>1.2.1</td>
<td><a href="http://www.unidata.ucar.edu/packages/netcdf-perl">http://www.unidata.ucar.edu/packages/netcdf-perl</a></td>
</tr>
<tr>
<td>Unidata UDUNITS</td>
<td>1.11.7</td>
<td><a href="http://www.unidata.ucar.edu/packages/udunits">http://www.unidata.ucar.edu/packages/udunits</a></td>
</tr>
</tbody>
</table>

To build the system, a FORTRAN compiler allowing promotion of all REAL variables to DOUBLE PRECISION is required. Typically, gcc does not provide this capability. Hence, in the development of the prototype MDSS, the Portland Group compiler was used for this purpose. This compiler costs approximately $500. The only program requiring this compiler is the road temperature model SNTHERM-RT and the bridge temperature model SNTHERM-RTB. An alternative to purchasing the Portland Group compiler would be to modify all the SNTHERM source code to convert REAL variables to DOUBLE PRECISION. The MDSS Release 3 CD contains a SNTHERM executable. Thus if one is running the system on a Red Hat UNIX platform, this compiler is unnecessary.

5.2.6 Inter-Process Communication

MDSS inter-process communication is done through the file system. Each process obtains its inputs solely through filenames and UNIX times specified on its command line. Many of these files specified are data files generated by upstream processes; many are static binary data files. The remainder are configuration files that contain the list of sites and forecast variables to be processed.
Each process reads its input data from the files on its command line, processes it, and writes its output to a file. The output file name is specified on the command line. The output file name’s format is specified in a file named on the command line.

The interface between the display and upstream processes is a client-server relationship. The display is the client; its requests are handled by a web server and processed using CGI scripts. The CGI scripts parse the requests and obtain the requested data. These data are then returned to the display.
An overview of the MDSS data flow is given in Figure 5.1.

![Diagram of MDSS data flow]

Figure 5.1. Overview of MDSS data flow.

### 5.2.7 Data Ingest

Several types of live data\(^1\) are required by the system. These data are all generated and disseminated from external sites and received through a connection to the Internet, or optionally via a NOAAPort system. Various types of numerical prediction model data (gridded or statistical) are used in creating the forecasts, and observation data are required for creating empirical relationships with the forecasts. These data are vital for the system to be able to make ‘tuned’ weather and road condition forecasts as described in Appendix A.

The live data used by the 2004 MDSS system are listed below.

---

\(^1\) Live data refers to data that flow into the system shortly after they are measured or obtained.
• NWS Eta model, 00, 12Z runs
• NWS GFS model, 00, 12Z runs
• NWS MAVMOS, 00, 06, 12, 18Z runs
• METAR observations
• DOT RWIS observations (environmental and road condition) from FSL MADIS
• FSL Ensemble models (MM5, WRF), run hourly
• FSL RUC, 00, 06, 12, 18Z runs

Figure 5.2 provides a schematic of the MDSS data ingest subsystem.

![Data Ingest Subsystem Diagram](image)

**Figure 5.2. Data ingest subsystem data flow diagram**

### 5.2.8 Road Weather Forecast System (RWFS)

Based on DICAST™ technology, the RWFS creates the weather forecast time series data required to drive the RCTM and provides the weather data presented by the display. The RWFS is tasked with ingesting reformatted meteorological data (observations, models, statistical data, climate data, etc.) and producing meteorological forecasts at user defined forecast sites and forecast lead times. The forecast variables output by the RWFS are used by the RCTM to calculate the road surface temperature and to determine a suggested treatment plan. In order to achieve this goal, the RWFS generates independent forecasts from each of the data sources using a variety of forecasting techniques. A single consensus forecast from the set of individual forecasts is generated at each user defined forecast point based on a processing method that takes into account the recent skill of each forecast module. This consensus forecast is nearly always more skillful than any component forecast.
Due to the proprietary nature of the core technology used in this subsystem (DICAST™), a highly detailed software description is not provided and only object code is being provided for this module. A high level description of the subsystem is available in Appendix A and a data flow diagram is provided in Figure 5.3.

---

2 The DICAST™ system was developed with non-public funds prior to the MDSS project. A license from the UCARF is required to obtain this code. Note, the prototype MDSS does not require this specific module to operate, but a suitable replacement containing time series weather prediction data is required.
Figure 5.3: An overview of the Road Weather Forecast System data flow used in 2004. Each Forecast Module (FM) generates a forecast. The Integrator process combines these into a consensus, final forecast.
5.2.9 Road Condition and Treatment Module (RCTM)

The RCTM ingests and processes environmental forecast data to predict the road surface temperature and snow depths at all forecast sites and lead times. Using the meteorological forecast data and the pavement temperature data, a predicted mobility index is calculated along with a treatment plan. For prototyping purposes, only a small number of forecast sites have been selected per highway route, and weather and road condition data are only processed for those sites. In an operational version of the system, it is anticipated that the users would identify regularly spaced and/or strategic locations along winter maintenance routes for which they want decision support and the system would be configured to generate treatment guidance for those locations. The output of the RCTM is passed to the display.

For this release, a different treatment plan is calculated for each highway forecast point. The treatment plans may vary spatially and no approach is currently in place to resolve any potential discrepancies between adjacent sites. These discrepancies may be partially due to differing subsurface structures of the roadways at adjacent points. For example, road conditions on a bridge can be very different from conditions a few hundred meters away. The RCTM, unlike the RWFS, is implemented as a single process. While similar at first glance, there are significant differences between the two subsystems that influence this decision. This is mainly due to feedback effects of plowing and chemical applications on the road. The treatment plan generation algorithm requires a time series of pavement temperatures and meteorological variables to determine the appropriate first treatment (time and type). Once that treatment has been implemented, the time series of pavement temperatures beyond that treatment time is invalid because of the sensitivity of the road temperature to the snow cover. The snow on the road at the time the pavement temperature time series was calculated will not exist (in the same state) after a plowing and chemical treatment. The RWFS design of multiple processes is effective in a forward processing flow system. However, the complex inter-module interactions led to a decision to implement the RCTM as a single process.

The driver process handles one site at a time. After a road temperature time series is generated, a treatment plan is developed. If the plan indicates a treatment should be applied, the processing of the site restarts from the first treatment time. The road surface state and meteorological conditions are modified to reflect the treatment action before regenerating another road temperature time series. In the MDSS it is assumed that plowing will always take place as part of a chemical treatment or abrasive application. This changes the road surface state by removing most of the existing snow.

If the Treatment Module recommends a chemical treatment, the suggested chemical amount is applied. The chemical algorithm then calculates the dissipation and effectiveness of the chemical based on time, traffic, and precipitation. The treatment impacts the road state and a new road temperature time series is calculated starting from the treatment time. This iterative process continues until no treatment is required in the remaining time until the end of the forecast period. See Figure 5.4.
The RCTM has been designed to determine road conditions and treatment plans in three main modes. The RCTM processing modes are as follows:

1) No treatment
2) Recommended treatment
3) User-defined treatment

The first mode assumes that no treatment will take place. This provides the user with information on the state of the road if nothing is done. The second mode generates a recommended treatment plan using the output from the coded rules of practice module (see Appendix E). The third mode generates road conditions based on a user-defined treatment plan(s). This third mode is used both for user-driven road condition calculation requests based on specified treatment plans (i.e. what-if’s) and for calculations of the road conditions under the garages’ current treatment plans. As part of its interaction with this system, each garage must regularly keep the system informed of its plan of action. These (user-specified) current treatment plans are stored on disk and used in these calculations.

5.2.10 Forecast Data to Display Reformatter

The data generated by the RWFS and RCTM are similar in format. Output data from both of these subsystems need to be passed to the MDSS display. This reformatter process ingests a list of variables required by the display, extracts those data, and writes an output file in the format required by the display.
5.2.11 Display – Forecast System Interface

The display system interfaces with a server to obtain its data. These interactions are handled by a web server, which has access to the MDSS forecast system data store. The web server invokes CGI scripts to handle the requests generated by the display. These requests are a mixture of polled and user-driven requests. See Figure 5.5.

![Figure 5.5. Overview of MDSS display client-server data flow.](image)

The display obtains the current time, data files and/or their status, and user permissions through this interface. It also uses the interface to pass user-defined treatment plan definitions to the server and return the resultant road condition to the display. For users in the DOT sheds with permissions to set current treatment plans, the display uses the interface to update the currently selected plan for a route when the user changes the selected treatment plan.

5.2.12 Functional Prototype Display

The display system provides an interactive display of data generated by the other MDSS components. The display system provides an interactive weather alert map, shows time series graphs of weather and road condition forecast products, shows recommended road treatments, and provides a mechanism for users to perform ‘what-if?’ scenarios with different road treatments.
6 SYSTEM OPERATIONS

6.1 Process Invocation

6.1.1 Schedule Driven Forecast System

Processes in the MDSS forecast system run on a schedule. The schedule is determined by examination of typical data arrival times and process run times. As much of the processing is sequential, it is necessary to allow sufficient time between process invocations to ensure that the preceding process will have finished. On the other hand, these inter-process scheduling gaps should be as short as is reasonable to ensure timelier end-to-end processing time.

Rather than using a custom scheduler, the MDSS uses the UNIX system scheduling utility cron. This flexible utility is documented online in the UNIX man pages. One drawback of using cron is its temporal resolution. The finest granularity available using cron is one minute. For the MDSS system, this is not a particularly oppressive restriction since the lead times the end user expects to use are more on the order of several hours. The trade-off of less custom development is made in exchange for slightly more latency in the system output.

6.1.2 Glue Layer

The knowledge of the file system layout lies within a relatively small piece of code known as the "glue layer". The glue layer's responsibility is to stick the system together. It creates each process's command line through its knowledge of the file system and other system resources.

When cron determines that it is time to take an action, it starts the executable listed in the schedule. For MDSS, this is typically a python super-script whose job is to start and oversee the execution of the process. This process sets a timer to ensure that the process it invokes finishes within a reasonable length of time. The super-script then invokes another python script specific to the scheduled task.

The job of the python script associated with the scheduled task is to determine the command line arguments for the scheduled C/C++ module's execution. It does this through its knowledge of the layout of the file system. There exist text-based "index files" at strategic locations within the file system that contain the name of the most recently generated file of a certain type. The python script reads the appropriate index files to obtain the required input file names. Usually, these file names will go directly onto the command line. Other times, the python script will create a configuration file containing a list of the file names. The configuration file's name would then go on the command line. The python
script also knows the location of the static files that are required by the C/C++ module. Their file names are also gathered and put on the command line. The python script may also query the system to obtain the current time.

This python script then executes the command line it has created. The process's return code is processed to determine whether it was successful in completing its task. Upon success, the script then ensures that the index files are updated with the new file names. If the process does not run to a successful completion, the index files are not updated, and other processes will never use the newly created file.

### 6.1.3 Treatment Update Network Layer

Processes may also be user-request driven and hence unscheduled. These requests are initiated by user actions in the display application, and then handled by a web server that invokes CGI python scripts. The CGI scripts parse the input parameters and attempt to satisfy the request. These python and perl scripts may either consult the file system or invoke other scripts or C++ based programs to obtain the results. An example of the latter would be a user-defined treatment plan request. In this case, the python script creates or gathers a number of configuration files, invokes the road condition module, monitors its progress, and returns the output.

### 6.2 Display Invocation

The Display system can be invoked through Java WebStart from Sun Microsystems. Through the use of WebStart, the user is able to either click on a web link in an html document, or start the application from the desktop. The WebStart invocation requires a connection to the Internet. For more information on WebStart, see section 16.

### 7 PROCESS ERROR HANDLING

#### 7.1 Data Logging

##### 7.1.1 Log Files

The MDSS Forecast Subsystem scripts and processes generate output indicating their progress and exit status. This textual output is written to MDSS system log files. Examination of these log files allows the system administrator to easily verify the status of each completed process and identify where problems occurred.
All messages are logged to files in the MDSS file system, with the exception of TUNL messages which are sent to the web server’s log file(s).

### 7.1.2 Log File Organization

Three sets of log files are maintained for each process. One file is associated with the python super-script to record its actions. A second log file is associated with the python script that creates and executes the command line. The command line that is actually executed is written to this log file. This allows a developer to re-create and debug the problem. The third log file contains output from the C/C++ module itself.

### 7.1.3 Log File Hierarchy

The first two sets of log files are each generated by a python script. They are responsible for logging information about their internal status as well as monitoring and reporting on their children's status. The third log file only reports on its internal status since it has no children.

Each script or process is required to write log messages indicating that it has started. It must also indicate if it has started another process and the command line associated with that invocation. It also must report on the status code returned by its child. Finally, it must log that it is done and report its status code. Specific formats exist for reporting starting and ending status. Searching the files for bad status messages is an easy means for finding problems.

### 7.1.4 Log File-Naming Conventions

The log files are all located in a log directory. For each C/C++ executable, one set of the three log files is created every day. The date associated with the files is part of the log files' names. The logging information for all runs of a particular executable during a given day is contained in these three log files.

The names of the log files differ only in their suffix. The python super-script's log file has an .epl suffix. The second python layer's log file has a .pyl suffix for python layer. Finally the C/C++ executable's log file has an .asc extension. As an example, the log file names for the RWFS integration process, int_fcst, for 13 December 2001 would be:

```plaintext
int_fcst.20011213.epl
int_fcst.20011213.pyl
int_fcst.20011213.asc
```
7.1.5 Executable’s Log File Messages

The log files written by the executable have standardized output. Each line in the file starts with the time the log message was written to the file. The next word on any line in the file is required to come from the following set which describes the function of the log message.

- **Starting**: Required at the start of the program.
- **Info**: Used for informational messages.
- **Warning**: Indicates non-fatal error conditions.
- **Error**: Indicates that a fatal error occurred. Explanation of the error must be provided.
- **Ending**: Required last statement before exiting. Exit status should also be indicated.

These standardized messages allow easy monitoring of the log files. Searching for the string “Error” identifies all gracefully exiting failures in the system. Searching for unmatched “Starting” and “Ending” pairs can point to non-graceful exits like segmentation violations.

7.2 Error Handling

7.2.1 Index Files

Index files exist in key locations throughout the file system. These files keep a list of all the files created by successfully completed processes. If an error occurs in the processing, the current output file which contains untrustworthy data is not registered in the index file. In this case, the file remains on the disk but is not visible to the system. The python glue layer only looks for file names listed in an index file when trying to determine the input files to be used on the next command line. Therefore, processes which fail have the same effect on the system as if they were never invoked.

7.2.2 Handling Old Data

If a process fails and the downstream process requires input of that type, the processing chain could fail. To avoid this possibility, each process must be able to handle data files that may not line up temporally with the other input data files. In this way, if a process failed, the output file from an earlier run of that process would be used to provide the required data. Though this data is older (i.e., not the most recent forecast run), it is still generally adequate for use in creating a reasonable forecast since the system utilizes multiple sources for forecast data. Hence, the system will use data from a previous forecast run as long as it is valid for the appropriate time. Clearly, if the span of data in the old file does not overlap enough with the current process’ time window, problems may ensue.
However, usually these problems can be detected and addressed early enough for an operator to effectively deal with the situation.

A real-time system monitor is not being provided as part of the MDSS prototype. It is expected that a system monitor would be developed and implemented as part of an operational MDSS system.

### 7.2.3 Failures in Client-Server Interactions

For a variety of reasons, problems may arise in the interactions between the display (client) and the server. Missing data or other system problems may result in a FAILURE code being returned to the client application when the server is accessed. Latency or a failure in the communication link may cause the client to never receive a response from the server. Script or process failure within the server’s processing could also cause a lack of response from the server. The goal of the MDSS system is to handle all of these failure cases robustly.

Client-server failure cases that the MDSS system attempts to handle are those that result in a return code of FAILURE from the server communications (TUNL) layer. These include everything from missing data to calls to the RCTM module which time out due to a processing error. Every script in the TUNL layer contains logic to return the FAILURE code if it encounters unexpected circumstances such as missing data or system calls that do not return in a reasonable length of time. These scripts are reasonably well tested in regard to returning FAILURE to the client display.

Failure cases that are currently unhandled in the display are those that result in a partial response from the TUNL layer. For example, if data is being delivered to the display from the TUNL and the data stops mid-stream, the display will continue to wait for the data, indefinitely. These cases are extremely rare and usually indicate a problem on the web server.

### 8 INTERPROCESS COMMUNICATION

#### 8.1 File System Communication

MDSS Forecast Subsystem inter-process communication is done through the file system; that is, data flows from one program to another by reading and writing files. Multiple simultaneous readers of the same files are allowed. Index files are used to ensure that a file is ready for use by the rest of the system.

Each process obtains input data generated upstream solely through filenames specified on its command line. The process reads this input data from the specified files, processes it, and writes its output to a file. The output filename is also specified on the command line.
The format of the output file is specified in a file whose name is specified on the command line.

### 8.2 File Formats

Files within the MDSS are in text or binary format. The netCDF format is the only binary format used internally within the MDSS. The netCDF format is a standard developed for scientific applications. It is widely used in the meteorological community. Information on the format can be obtained at:

[www.unidata.ucar.edu/packages/netcdf](http://www.unidata.ucar.edu/packages/netcdf)

The netCDF format data files use the .nc extension. For reference information on software packages that may be used for manipulating or displaying netCDF data, see:

[http://www.unidata.ucar.edu/packages/netcdf/software.html](http://www.unidata.ucar.edu/packages/netcdf/software.html)

#### 8.2.1 Typical MDSS netCDF Formats

NetCDF is an extremely flexible self-defining format. Within the MDSS, only simple instances of the format are used. Typically, complex structures are avoided. Instead, the data is broken out by variable. It is stored in multidimensional arrays much like the data structures typically used in FORTRAN programming. The array dimensions are hard-wired within the format. Typically the only dimension that might need to be changed when implementing this system would be the `max_sites` dimension. This should be adjusted to be at least as large as the number of forecast sites.

ASCII text files that describe the format of the binary netCDF output files, exist within the system. These files, called Command Description Language (CDL) files, have a .cdl extension. These are the files which need to be edited to reflect forecast site list size when configuring the system.

#### 8.2.2 Text File Formats

Formats for text files are ad hoc and described within this document.
8.3 Display Request Interprocess Communication

The display formats its requests in a typical CGI-bin syntax. That is, web server address and the CGI script name are followed by a question mark delimited argument list. These arguments are parsed by the appropriate CGI python script.

9 CONFIGURATION FILES

9.1 Configuration File Overview

The MDSS system uses configuration files in various formats to tailor the ultimate output of the system. Most configuration files are plain-text ASCII files, but some are static files in netCDF format. In general, the default configuration files will not need to be modified. An exception to this is if the user changes the forecast site list. In this case, modifications to a number of other configuration files (and possibly CDL files) will be required to generate the new static configuration files. See the installation guide that is being provided as part of this release (on CD) for more information.

9.2 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. The site list files are text files which 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)
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.

10 REDUNDANCY

The decision on whether to provide redundant components in the system is driven by two main factors:

- The consequences of a failure
- The cost of providing redundancy

Since the MDSS prototype is a prototype and not considered a safety-critical system, a moderately conservative approach to redundancy was considered.

Most of the input data required by the MDSS can be received in a timely fashion over the Internet with typical T1 bandwidth (~1MB/sec). The communications network infrastructure represents a large recurrent cost, with monthly payments to the communications vendor company. It is therefore not cost effective to plan on redundant Internet communications links. However, if desired, the NCEP data can be obtained through a NOAAPort system, which provides a satellite downlink capability. Beyond the cost of the NOAAPort Receiving Dish, there is no recurrent cost. Between these two communications methods, loss of incoming data should be extremely rare.

A server will serve the output forecast and treatment plan data over the Internet to the end users’ displays. It is not foreseen to be cost effective to provide a duplicate Internet connection. Operational versions of the system could utilize Internet or Intranet network communications.

Beyond the data ingest processes, complete redundancy can be obtained by having two duplicate processing systems (hardware/software). Unless an extremely large site configuration is used, the cost of these redundant machines should be relatively minor. Software to detect a failure and switchover to the backup machine would have to be developed.

Components of the LAN, e.g. routers and switches, have been found to be extremely reliable and it is not considered necessary to have complete redundancy. However, one extra unit of each type should be configured and ready to be swapped should failure occur.

For the machines in the system, sufficient spare equipment should be kept on hand to allow for quick replacement of the faulty machine or component.
11 DATA INGEST

The source code (scripts) for the data ingest subsystem can be found on the RWFS CD in the /scripts/perl directory.

11.1 Identification

N/A

11.2 Type

Subsystem, i.e., collection of interconnected processes.

11.3 Purpose

To ingest real-time data and decode it into a format usable by the RWFS system.

11.4 Function

The Local Data Manager (LDM) from Unidata is used to acquire real-time data consisting of observations (METARs and RWIS or state DOT reports), NWS Model Output Statistics (MOS) and mesoscale model GRIB products from FSL. The LDM acquires these data and runs them through decoders to put them into netCDF format. These METAR and MOS decoders can be found on the RWFS CD in the /scripts/perl directory. They are metar2nc (Unidata decoder modified for local use) and mos2nc.

The other component of the ingest subsystem is model data acquisition via ftp. A set of scripts ftp's the model data from the NWS ftp server. These data consist of the GFS (AVN) and Eta model runs at 00 and 12 UTC. These data are acquired using ftp and are initially in GRIB format. The ftp scripts convert the data into netCDF format using the grib2onc program (written by Unidata and modified for local use), which can be found on the RWFS CD.

11.5 Dependencies

A connection to a NOAAPort (local or external over the Internet) and FSL Meteorological Assimilation Data Ingest System (MADIS) data feed is required by the MDSS prototype to ingest and decode observational and MOS data. For simplicity, the MDSS obtains RWIS
data from participating state DOTs through the FSL MADIS data feed. For more information on MADIS, see:

http://jailbird.fsl.noaa.gov/MADIS/

A connection to the Internet is required to transfer NWS model data from the NWS ftp server and acquire the ensemble model data from FSL. The LDM and ftp scripts must be properly configured to ingest and decode data into a usable format in a place the rest of the RWFS can access. These configuration files can be found on the RWFS CD in the /etc and /scripts/perl directories.

11.6 Interface

11.6.1 Command Interfaces

11.6.1.1 LDM Command Interface

See the LDM documentation in the RWFS installation guide, which is provided as part of this release (on the RWFS CD in /docs/rwfs_install_guide.doc), for information on starting the LDM system. Additional information on the LDM can be obtained on the Unidata website at:

http://my.unidata.ucar.edu/content/software/ldm

11.6.1.2 FTP Script Command Interface

The FTP scripts are run from cron at frequent intervals to "poll" for new data. A sample crontab file containing entries to do this polling is provided in the /etc subdirectory of the RWFS CD. The perl script ftp_model_data.pl is run in combination with a configuration file to retrieve the model data.

% ftp_model_data.pl [-n] [-c] [-s] model_config.pl refhour [YYYYMMDD]

Options:

- $n$ Do NOT decode files to netCDF
- $c$ Do NOT concatenate to a single bzipped file
- $s$ Create symbolic link for alternate naming syntax

model_config Path to file defining model-specific elements
refhour Model reference hour (e.g.; 00, 12)
YYYYMMDD Optional date to retrieve instead of today

The ftp_model_data.pl script and configuration files for each model can be found on the RWFS CD in /scripts/perl.
11.6.2 Configuration Files

11.6.2.1 LDM Configuration Files

Two main configuration files control the LDM. These files control what data is received, where it comes from, and what is done with it when it arrives. The file ldmd.conf is a text file which specifies the LDM components to run, what data feeds to get, and from where. The file pqact.conf is a text file which describes the actions which will be performed on arriving products. Sample files for each of these are provided in the /etc subdirectory on the MDSS RWFS CD. See the LDM documentation in the RWFS installation guide for more information.

11.6.2.2 FTP Script Configuration Files

The ftp script mentioned above uses a configuration file on the command line to determine what model data to get. This file is executable perl code, which defines several variables used in the ftp script. See the crontab.ldm file example located in the /etc directory of the RWFS CD.

11.6.3 Input

Input data consists of coded ASCII text from the NOAAPort data feed, GRIB messages from the NWS FTP server and FSL grid feed, and netCDF files from FSL's MADIS data feed, which is the RWIS data (atmospheric and road condition) from the DOT.

11.6.4 Output

Output files are in netCDF format usable by the RWFS system.

11.7 Processing

The LDM is started upon system startup and remains running as a set of daemon processes. As data arrives, it is filed and decoded into netCDF format as specified in the pqact.conf file. A number of external netCDF decoders are spawned to decode certain data.

The ftp scripts are run at timed intervals to check on the existence of new model data. These scripts poll the NWS ftp server and exit if no new data are available. If new data are available, the new data are ftp'd to the local host and then decoded into netCDF format.
12 ROAD WEATHER FORECAST SYSTEM (RWFS)

Note, due to intellectual property restrictions, the Road Weather Forecast System (RWFS) code and documentation are not contained on the public domain MDSS Release-3 CD. The RWFS, which is based on DICAST™ technology, requires a license from the University Corporation for Atmospheric Research Foundation (UCARF). To obtain a license for obtaining these materials, the reader is directed to:


After execution of the license agreement, a CD containing the RWFS object code and documentation will be provided.

All portions of section 12 pertain to the RWFS CD only.

12.1 Identification

N/A

12.2 Type

Subsystem, i.e., collection of interconnected processes.

12.3 Purpose

The RWFS produces weather forecast data at a number of locations. The forecasts are made site by site for each site supplied to the subsystem. The weather forecasts have hourly resolution out to 48 hours. The forecast variables are predictions of commonly observed meteorological surface measurements. These forecast weather conditions are passed on to the display to be presented directly and to the RCTM where the road conditions are predicted for each user-defined roadway site.

12.4 Dependencies

A number of static configuration files are required by the RWFS. These files are located under the /mdss_data/static_data directory on the RWFS CD and should not need modification. Each module within the subsystem has been designed to work whether or not the latest data it requires has arrived. That is, if a particular type of data does not appear on any particular day, the system will find the most recent data of that type and try to make its forecast using the older data set. For best results, data should arrive reliably. However, the system can handle missing or latent data. If too much data are missing from too many data
sources, it is possible that missing data values will appear in the output files. This may compromise the ability of downstream processes to successfully accomplish their processing. A system/process monitor should be developed for an operational system.

**12.5 Interfaces**

Due to intellectual property protections, the interfaces to the processes within this subsystem are not described in detail here. This subsystem should be viewed as a ‘black box’ that will operate if properly installed and configured. The site list file will need to be created. The format of this file is described in section 9.2 of this document.

**12.5.1 Input**

Several types of data are used by the RWFS. The data are decoded into netCDF format by the data ingest subsystem’s decoders. Each decoder uses a CDL template file that describes the variables that will be decoded to the output file. These files are provided in the /etc directory of the RWFS CD. For the 2004 winter demonstration, the file names are:

- gfs_003.cdl: Template for GFS model data
- eta_212.cdl: Template for Eta model data
- mm5eta.cdl: Template for MM5 model with Eta initialization
- wrfeta.cdl: Template for WRF model with Eta initialization
- metar.cdl: Template for METAR data
- mav_mos.cdl: Template MAVMOS data
- rucx_252.cdl: Template for RUC model data

**12.5.2 Output**

The RWFS outputs data at hourly resolution. Forty-eight hours of forecast data are required to run the RCTM. These 48 hours should be contained within the 4 days of hourly weather data contained in the RWFS output file. The format of the output data is the same as the input weather data for the RCTM. The CDL file describing that format can be seen in section 13.6.4.1 of this document.

**12.6 Processing**

An overview of the implementation of major RWFS processes is provided below.

**12.6.1 Model Data Empirical Relationship Subsystem**

**12.6.1.1 Regressor Calculation Processes**
These processes extract site-specific data from the gridded raw model data. Some variables are not explicitly predicted by the models and are derived here (e.g., relative humidity).

12.6.1.2 DMOS Empirics Processes

These processes look for relationships between the observations and the regressor data. These relationships are stored for use by the DMOS Forecast Modules.

12.6.2 Forecast Modules

12.6.2.1 DMOS Forecast Modules

These forecast modules take the DMOS empirics file and apply the relationships stored therein to the latest regressor data in order to create its forecast.

12.6.2.2 NWS MOS Forecast Modules

These forecast modules use the forecast data provided at the MOS sites to create their forecasts. The forecasts at the MOS sites are passed-through and the forecasts at other sites are created by “smart” interpolation techniques.

12.6.3 Climatology Forecast Module

This forecast module makes its forecast based on climatological data. If no climatological data are available for a particular site, a forecast consisting of missing data is produced. A climatology file for METAR sites is included with the RWFS CD. The compressed file is named /mdss_data/st/static_data/climate/dicast_climatology.nc.gz.

12.6.4 Forecast Integrator

The integrator combines the forecasts produced by each of the forecast modules. It creates consensus forecast by doing a bias-corrected confidence-weighted sum of the forecast modules outputs.

12.6.4 Forecast Integrator Empirics

The integrator empirics process modifies the weights assigned to each forecast module. The weight changes are dependent on the relative errors of each of the forecast modules. A
different set of weights exists for every forecast variable at each forecast lead-time at every site.

12.6.5 Non-Verifiable Data Extractor

This process extracts forecast variables required by the RCTM, but not commonly observed. These variables are thus non-verifiable and the forecast process cannot be tuned.

12.6.6 Post Processor

This suite of processes merges the integrated forecasts and the non-verifiable forecast data. It also performs basic quality control procedures on the outputs. Spatial and temporal interpolation may also be performed. For observed variables, a Forward Error Correction (FEC) process is used to help ensure that the “current” and near term forecasts better match the recent observations. In addition, several derived variables are produced which are required by the RCTM and the display.

13 ROAD CONDITION AND TREATMENT MODULE (RCTM)

The source code for this section can be found on the public domain CD in the /src/apps/road_cond/ directory. The binary file can be found in the /bin directory.

13.1 Identification

road_cond

13.2 Type

Process

13.3 Purpose

The purpose of the RCTM is to produce road condition forecasts and treatment plan related data. The road conditions depend on the treatments applied to the road. Three treatment options are available, 1) no treatment, 2) system suggested treatments from the Rules of Practice module, and 3) user-defined treatments. These forecasts are made site-by-site for each site in the list supplied to the program. The forecast road condition surface conditions are comprised of surface temperature and snow depth. Treatment related data produced are the treatment plans developed or used in the program run and a time series of the chemical
concentration on the road. These treatment plans and road conditions are of primary interest to the end user and are made available to the display.

13.4 Function

The RCTM reads weather condition, road state, treatment option, and site configuration data. These data are processed to produce the road condition output data of interest to the end users. Its output data files are made available to the Forecast Reformatter for preparation before being passed to the display.

More information on the Road Temperature, Net Mobility, Chemical Concentration, and Rules of Practice modules are provided in the appendices.

13.5 Dependencies

The RCTM requires weather condition data produced by the RWFS. These weather data must be valid, i.e. non-missing, in all relevant fields at every forecast site from the road condition forecast starting time through the ending time (48 hours later).

Road characteristics (e.g., pavement type and depth, subsurface type and depth, etc.) for each site are required. These characteristics must be properly parameterized for use by SNHERM. Additionally, road segment specific parameters such as idealized traffic level, number of lanes, route length, and typical route treatment times must be specified for each site/segment.

The RCTM may initialize its subsurface temperature profile from a previous run of the road condition module. If so, that run must have been successful and have no missing subsurface temperature data for any site at the start time of the current road condition forecast run. If a subsurface temperature profile is provided, the subsurface structure may not have changed from the previous run. That is, the subsurface parameterization cannot change if previously computed temperatures are to be used. Also, no new site can have been added since it will not appear in the previous run’s output file. If changes of these types have been made, the RCTM must be run at least once without a “previous run” data file. If no previous run’s subsurface data are available, the soil subsurface temperatures from the Eta model are interpolated through the various subsurface layers.

13.6 Interfaces

13.6.1 Command Interface

% road_cond forecast_time site_list bridge_site_list cdl_file materials_file site_layer_traffic_file chem_type treatment_option user_def_treatment_file
previous_road_cond_file rwis_obs_file weather_forecast_file output_file [-d debug_level] [-l log_file]

where:

- **forecast_time** is the UNIX time of the first road condition forecast to be generated. This will be truncated to the top of the hour.
- **site_list** is the name of the text file containing the list of sites to be processed.
- **bridge_site_list** is the name of the text file containing the list of bridge sites to be processed.
- **cdl_file** is the name of the CDL file describing the output format.
- **materials_file** is the name of the configuration file containing properties of materials typically used in road construction such as asphalt, concrete, aggregate base, etc.
- **site_layer_traffic_file** is the name of the configuration file containing information specific to a roadway segment. This includes both 1) the subsurface layers at each site and parameterization of those layers for use by SNTHERM, and 2) idealized hourly traffic levels, number of lanes, and length of the road segment.
- **chem_type** is the desired chemical to be used in treatment calculations:
  - 1 = NaCl
  - 2 = CaCl$_2$
  - 3 = MgCl$_2$
  - 4 = CaMg Acetate
  - 5 = K Acetate
- **treatment_option** is the desired treatment option:
  - 0 = No Treatment
  - 1 = Suggested Treatment to be generated from Rules of Practice
  - 2 = User-defined treatment
- **user_def_treatment_file** is the name of the file containing the user defined treatment plan. This argument is ignored if treatment_option is not 2. Commonly, “None” is used for this argument if treatment_option is not 2.
- **previous_road_cond_file** is the name of the output file from an earlier run of road_cond. A run from 3 or more hours earlier is typically used. The earlier run must be from within the previous 48 hours, and preferably within the last 12 hours.
- **rwis_obs_file** is the name of the text file containing the recent pavement and subsurface observations from the RWIS sites.
- **weather_forecast_file** is the name of the weather forecast netCDF output file from the RWFS.
- **output_file** is the name of the netCDF output file to be produced.
- **debug_level** is the level of debugging information to be output to the log file. The default level is 0.
- **log_file** is the file to which log output should be written.
13.6.2 Start Script

The roadCond program is started by a python script, which gathers the names of the appropriate input files, static (configuration) files, and then runs the program. Treatment options and static data files are set within python "wrapper" scripts called ep_rc_no_tmt.py, ep_rc_rec_tmt.py and ep_rc_cur_tmt.py for no treatment, recommended treatment, and current (selected) treatment, respectively. Each of these scripts sets arguments and then runs a script called run_proc.py, which in turn actually executes the roadCond program on the real-time data.

RoadCond can also be invoked by the user-defined python script in the TUNL. This script constructs the command line, runs the program, and passes the output file back to the display.

13.6.3 Configuration Files

13.6.3.1 Site List Configuration File

This file contains information on the RCTM output sites. The format of this file is described in section 9.2 of this document.

13.6.3.2 Bridge Site List Configuration File

This file contains information specifying which of the output sites are bridges. The format of this file is the same as for the site list file and is described in section 9.2 of this document.

13.6.3.3 Materials Configuration File

This file contains properties associated with different materials commonly used in road construction. This file should not have to be modified unless a new material type needs to be added or it is determined that these generic material properties differ significantly from those in use in the particular state. Once created, this file’s name is passed to the RCTM as a command line argument.

The CDL file describing this netCDF file format is:

```plaintext
netcdf materials {
  dimensions:
    max_material_type = 10 ;
    mnemonic_len = 16 ;
  variables:
```
int type;
    type:long_name = "cdl file type";

int num_material;
    num_material:long_name = "number of actual material types";

int material_code(max_material_type);
    material_code:name = "material code id";

float quartz_content(max_material_type);
    quartz_content:long_name = "quartz_content";
    quartz_content:units = "decimal";

float roughness_length(max_material_type);
    roughness_length:long_name = "roughness_length";
    roughness_length:units = "meters";

float latent_heat_transfer(max_material_type);
    latent_heat_transfer:long_name = "latent_heat_transfer";
    latent_heat_transfer:units = "W/m^2";

float sensible_heat_transfer(max_material_type);
    sensible_heat_transfer:long_name = "sensible_heat_transfer";
    sensible_heat_transfer:units = "W/m^2";

float convective_latent_heat(max_material_type);
    convective_latent_heat:long_name = "convective_latent_heat";
    convective_latent_heat:units = "W/m^2";

float convective_sensible_heat(max_material_type);
    convective_sensible_heat:long_name = "convective_sensible_heat";
    convective_sensible_heat:units = "W/m^2";

float fractional_rh(max_material_type);
    fractional_rh:long_name = "fractional_rh";
    fractional_rh:units = "decimal";

char mnemonic(max_material_type, mnemonic_len);
    mnemonic:long_name = "mat type name";

float dry_density(max_material_type);
    dry_density:long_name = "dry_density";
    dry_density:units = "kg/m^3";

float bulk_dry_density(max_material_type);
    bulk_dry_density:long_name = "bulk_dry_density";
    bulk_dry_density:units = "kg/m^3";

float heat_capacity(max_material_type);
    heat_capacity:long_name = "heat_capacity";
    heat_capacity:units = "J/kg-K";

float thermal_conductivity(max_material_type);
    thermal_conductivity:long_name = "thermal_conductivity";
    thermal_conductivity:units = "W/m-K";

float coarseness(max_material_type);
    coarseness:long_name = "coarseness";
    coarseness:units = "1=coarse, 0=fine";

float plasticity(max_material_type);
    plasticity:long_name = "plasticity";
    plasticity:units = "decimal";

float albedo(max_material_type);
    albedo:long_name = "albedo";
albedo:units = "dimensionless";
float emissivity(max_material_type);
emissivity:long_name = "emissivity";
emissivity:units = "dimensionless";
float bulk_water_density(max_material_type);
bulk_water_density:long_name = "bulk_water_density";
bulk_water_density:units = "kg/m^3";
}

The SNTHERM-RT interface specification documents in this release, (/docs/SNTHERM/LAYERIN.DOC and /docs/SNTHERM/METIN.DOC), should be consulted for more information. These interface specification documents describe how to run SNTHERM-RT in a stand-alone mode. In addition, they provide more details on each of the above variables. CRREL should be consulted (see Appendix J for points of contact) if more specific information is required.

### 13.6.3.4 Site Layer and Traffic Configuration File

The Site Layer and Traffic Configuration file contains the subsurface characteristics at each forecast site as well as other site/segment characteristics (such as traffic levels, etc). The subsurface structure is broken down into several layers each with an associated material type (found in the materials configuration file). Note that layers are listed from the lowest layer up. Each layer is broken into a number of nodes that are used by the land-surface model SNTHERM-RT. Associated with each node is a node thickness. The depth of a node should be obtained by accumulating all the depths above that node. An algorithm to specify the nodal parameterization (number and thickness) for a given layer configuration is not yet available. It is anticipated that a nodal parameterization algorithm will be produced in 2004. In the meantime, please consult CRREL to obtain instructions on how to configure these data. Once created, this file’s name is passed to the RCTM as a command line argument.

This file is a text file. It must contain information on all the sites listed in the Site List configuration file. That is, the RCTM must be able to determine the characteristics for every site that it is tasked with handling. The configuration data is listed site by site. Sites should be sorted by site ID in increasing order. The format for a single site is given below. An example for one site follows the format description. A sample file with multiple sites can be found on the public domain CD at /mdss_data/rctm/static_data/config/site_layer_traffic.conf. Text on a line that appears after a # is treated as a comment.

```
SITE_ID: site_id # site ID number
NUM_LAYERS: num_layers # number of material types in subsurface
LAYER_MAT: layer_mat_1 layer_mat_2 layer_mat_3 ... # one layer code number per material type
NUM_NODES_PER_LAYER: num_node_1 num_node_2 ... # number of nodes in each layer
NODE_THICKNESS: thick_1 thick_2 ... # thickness of each node in meters
TRAFFIC: level_1 level_2 ... level_24 # traffic levels for each hour (start at midnight)
ROUTE_LENGTH: length # length of route (miles)
```
ROUTE_LANES: num_lanes # total number of lanes on segment
ROUTE_TREATMENT_TIME: treat_time # time required to treat route (minutes)

Note that there should be num_layers values provided after LAYER_MAT and NUM_NODES_PER_LAYER. Material types can be found in the Materials Configuration file, /mdss_data/rctm/static_data/config/layer.nc on the public domain CD. Note that this is a netCDF file. The thickness of each node should be listed after NODE_THICKNESS. Nodes are listed from the lowest (deepest) node upwards.

Recognized traffic levels are:
1 = Low (less than 250 vehicles per hour per lane)
2 = Medium (between 250 and 2000 vehicles per hour per lane)
3 = High (more than 2000 vehicles per hour per lane)

Here is a sample configuration for one site.

# IA 210 – Rural Primary
SITE_ID: 74449041
NUM_LAYERS: 4
LAYER_MAT: 90 92 94 1
NUM_NODES_PER_LAYER: 7 3 5 0
NODE_THICKNESS: 5 2 1 0.5 0.5 0.1 0.03 0.04 0.07 0.04 0.033 0.06 0.06 0.02 0.01
TRAFFIC: 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 2 1 1 1 1
ROUTE_LENGTH: 41.2
ROUTE_LANES: 2
ROUTE_TREATMENT_TIME: 167

13.6.4 Inputs
13.6.4.1 Weather Forecast Data File

The weather forecast data file is in netCDF format. Not all the available fields are read in. Fore_time must be 00Z of a particular day. The forecast valid times are one hour apart from that time. The 48 hours used by the RCTM will lie somewhere within this time period – not necessarily at the start of the input data. The data start time to be used by the process is determined by the command line argument forecast_time. This file’s name is passed to the RCTM as a command line argument. The file can be found on the public domain CD at /mdss_data/rctm/static_data/cdl/mesh_derive.cdl.

The CDL file describing this netCDF file format is:

```plaintext
netcdf post_process {
  dimensions:
    max_site_num = 150;  // number of locations
    days = 4;           // number of days
```
fc_times_per_day = 24;  // fcst times per day
daily_time = 1;
max_strlen = 10;       // max string length for location codes

variables:
    double creation_time;
    creation_time:long_name = "time at which forecast file was created";
    creation_time:units = "seconds since 1970-1-1 00:00:00";

double forc_time;
    forc_time:long_name = "time of earliest forecast";
    forc_time:units = "seconds since 1970-1-1 00:00:00";

int     num_sites;
    num_sites:long_name = "number of actual sites";

int     site_list(max_site_num);
    site_list:long_name = "forecast site list";
    site_list:_FillValue = -99999;

float   T(max_site_num, days, fc_times_per_day);
    T:long_name = "temperature";
    T:units = "degrees Celsius";

float   max_T(max_site_num, days, daily_time);
    max_T:long_name = "maximum temperature";
    max_T:units = "degrees Celsius";

float   min_T(max_site_num, days, daily_time);
    min_T:long_name = "minimum temperature";
    min_T:units = "degrees Celsius";

float   dewpt(max_site_num, days, fc_times_per_day);
    dewpt:long_name = "dewpoint";
    dewpt:units = "degrees Celsius";

float   cloud_cov(max_site_num, days, fc_times_per_day);
    cloud_cov:long_name = "cloud cover";
    cloud_cov:units = "percent";

float   prob_fog(max_site_num, days, fc_times_per_day);
    prob_fog:long_name = "probability of fog";
    prob_fog:units = "percent";

float   prob_thunder(max_site_num, days, fc_times_per_day);
    prob_thunder:long_name = "probability of thunder";
    prob_thunder:units = "percent";

float   cprob_rain(max_site_num, days, fc_times_per_day);
cprob_rain:long_name = "conditional probability of rain";
cprob_rain:units = "percent";

float cprob_snow(max_site_num, days, fc_times_per_day);
cprob_snow:long_name = "conditional probability of snow";
cprob_snow:units = "percent";

float cprob_ice(max_site_num, days, fc_times_per_day);
cprob_ice:long_name = "conditional probability of ice";
cprob_ice:units = "percent";

float prob_precip01(max_site_num, days, fc_times_per_day);
prob_precip01:long_name = "probability of precipitation, 1 hr";
prob_precip01:units = "percent";

float prob_precip03(max_site_num, days, fc_times_per_day);
prob_precip03:long_name = "probability of precipitation, 3 hr";
prob_precip03:units = "percent";

float prob_precip06(max_site_num, days, fc_times_per_day);
prob_precip06:long_name = "probability of precipitation, 6 hr";
prob_precip06:units = "percent";

float prob_precip24(max_site_num, days, daily_time);
prob_precip24:long_name = "probability of precipitation, 24 hr";
prob_precip24:units = "percent";

float qpf01(max_site_num, days, fc_times_per_day);
qpf01:long_name = "amount of precipitation";
qpf01:units = "meters";

float qpf03(max_site_num, days, fc_times_per_day);
qpf03:long_name = "amount of precipitation";
qpf03:units = "meters";

float qpf06(max_site_num, days, fc_times_per_day);
qpf06:long_name = "amount of precipitation";
qpf06:units = "meters";

float wind_u(max_site_num, days, fc_times_per_day);
wind_u:long_name = "eastward-component of wind";
wind_u:units = "meters per second";

float wind_v(max_site_num, days, fc_times_per_day);
wind_v:long_name = "northward-component of wind";
wind_v:units = "meters per second";

float visibility(max_site_num, days, fc_times_per_day);
visibility:long_name = "visibility";
visibility:units = "km";

float wind_speed(max_site_num, days, fc_times_per_day);
wind_speed:long_name = "windspeed";
wind_speed:units = "meters per second";

float wind_speed_mph(max_site_num, days, fc_times_per_day);
wind_speed_mph:long_name = "windspeed in mph";
wind_speed_mph:units = "miles per hour";

float wind_dir(max_site_num, days, fc_times_per_day);
wind_dir:long_name = "wind direction clockwise from north";
wind_dir:units = "degrees north";

float rh(max_site_num, days, fc_times_per_day);
rh:long_name = "relative humidity";
rh:units = "decimal";

float rh_pct(max_site_num, days, fc_times_per_day);
rh_pct:long_name = "percent relative humidity";
rh_pct:units = "percent";

float precip_rate(max_site_num, days, fc_times_per_day);
precip_rate:long_name = "precip (SWE) rate";
precip_rate:units = "mm/hr";

float precip_rate_inches(max_site_num, days, fc_times_per_day);
precip_rate_inches:long_name = "precip (SWE) rate";
precip_rate_inches:units = "in/hr";

float precip_type(max_site_num, days, fc_times_per_day);
precip_type:long_name = "precipitation type";
precip_type:units = "0=NONE, 1=RAIN, 2=SNOW, 5=ICE";

float precip_accum(max_site_num, days, fc_times_per_day);
precip_accum:long_name = "3 hr precip accumulation";
precip_accum:units = "mm";

float precip_accum_inches(max_site_num, days, fc_times_per_day);
precip_accum_inches:long_name = "3 hr precip accumulation";
precip_accum_inches:units = "inches";

float TempF(max_site_num, days, fc_times_per_day);
TempF:long_name = "temperature";
TempF:units = "degrees Fahrenheit";

float dewptF(max_site_num, days, fc_times_per_day);
dewptF:long_name = "dewpoint";
dewptF:units = "degrees Fahrenheit";
float snow_rate(max_site_num, days, fc_times_per_day);
snow_rate:long_name = "snowfall rate";
snow_rate:units = "mm/hr";

float snow_rate_inches(max_site_num, days, fc_times_per_day);
snow_rate_inches:long_name = "snowfall rate";
snow_rate_inches:units = "in/hr";

float snow_accum(max_site_num, days, fc_times_per_day);
snow_accum:long_name = "3 hr snowfall accumulation";
snow_accum:units = "mm";

float snow_accum_inches(max_site_num, days, fc_times_per_day);
snow_accum_inches:long_name = "3 hr snowfall accumulation";
snow_accum_inches:units = "inches";

float snow_accum_total(max_site_num, days, fc_times_per_day);
snow_accum_total:long_name = "snowfall accumulation since start of forecast";
snow_accum_total:units = "mm";

float snow_accum_total_inches(max_site_num, days, fc_times_per_day);
snow_accum_total_inches:long_name = "snowfall accumulation since start of forecast";
snow_accum_total_inches:units = "inches";

float prob_precip03_pct(max_site_num, days, fc_times_per_day);
prob_precip03_pct:long_name = "probability of precipitation, 3 hr";
prob_precip03_pct:units = "percent (0-100)";

float blowing_snow_potential(max_site_num, days, fc_times_per_day);
blowing_snow_potential:long_name = "blowing snow potential";
blowing_snow_potential:units = "index (0-3) (low-high)";

float blowing_snow_pot_vals(max_site_num, days, fc_times_per_day);
blowing_snow_pot_vals:long_name = "blowing snow potential values";
blowing_snow_pot_vals:units = "floating point (0-1) (low-high)"

float P_sfc(max_site_num, days, fc_times_per_day);
P_sfc:long_name = "Pressure at 2m above sfc";
P_sfc:units = "millibars";

float T_cb(max_site_num, days, fc_times_per_day);
T_cb:long_name = "cloud base temp";
T_cb:units = "degrees Celsius";

float T_bls(max_site_num, days, fc_times_per_day);
T_bls:long_name = "sub-sfc temperature";
T_bls:units = "degrees Celsius";

float T_lbls0(max_site_num, days, fc_times_per_day);
T_lbls0:long_name = "0-10 cm layer sub-sfc temperature";
T_lbls0:units = "degrees Celsius";

float T_lbls1(max_site_num, days, fc_times_per_day);
T_lbls1:long_name = "10-40 cm layer sub-sfc temperature";
T_lbls1:units = "degrees Celsius";

float T_lbls2(max_site_num, days, fc_times_per_day);
T_lbls2:long_name = "40-100 cm layer sub-sfc temperature";
T_lbls2:units = "degrees Celsius";

float T_lbls3(max_site_num, days, fc_times_per_day);
T_lbls3:long_name = "100-200 cm layer sub-sfc temperature";
T_lbls3:units = "degrees Celsius";

float snow_depth(max_site_num, days, fc_times_per_day);
snow_depth:long_name = "water equiv of accum snow depth";
snow_depth:units = "kg/m2";

float cloud_low(max_site_num, days, fc_times_per_day);
cloud_low:long_name = "low cloud layer amt";
cloud_low:units = "decimal";

float cloud_middle(max_site_num, days, fc_times_per_day);
cloud_middle:long_name = "middle cloud layer amt";
cloud_middle:units = "decimal";

float cloud_high(max_site_num, days, fc_times_per_day);
cloud_high:long_name = "high cloud layer amt";
cloud_high:units = "decimal";

float dlwrf_sfc(max_site_num, days, fc_times_per_day);
dlwrf_sfc:long_name = "downward long wave radiation flux at surface";
dlwrf_sfc:units = "W/m2";

float dswrf_sfc(max_site_num, days, fc_times_per_day);
dswrf_sfc:long_name = "downward short wave radiation flux at surface";
dswrf_sfc:units = "W/m2";

float albedo_sfc(max_site_num, days, fc_times_per_day);
albedo_sfc:long_name = "albedo at surface";
albedo_sfc:units = "percent";
}
13.6.4.2 Previous Road Conditions Forecast Data File

If available, the netCDF data file containing road conditions from a previous road cond run should be used. Otherwise the filename “None” should be used for the previous road conditions data file. The road subsurface temperature profile data, snow depth, and the state of the chemicals on the road are read in. The CDL file describing this netCDF file format is the same as the output file described in the next section. This file’s name is passed to the RCTM as a command line argument.

13.6.4.3 RWIS Observations File

If available, the observations from the previous hour should be provided. They are used to nudge the initial conditions of the subsurface profile towards the observations. This file is a space-delimited text file. Each observation in the previous hour from every RWIS should be included. Currently, the RCTM software assumes that every site’s observations appear in chronological order. This processing limitation should be addressed in the next release. The format is as follows:

```
site_id1  unix_time1  pavement_temp1  subsurface_temp1
site_id2  unix_time2  pavement_temp2  subsurface_temp2
...
```

Site_id is the site’s ID number from the site list file. The time is the UNIX time of the observation. The temperatures are in degrees Centigrade. The missing value for the temperatures is -9999. This will often be required if a site only reports a pavement temperature and not a subsurface temperature. All four fields are required. Line after line of observations should appear in the file.

13.6.5 Output

The road conditions data file is in netCDF format. Many derivative fields, e.g. units converted fields, are calculated for the display to lessen its complexity. Note that the output data starts at 00Z of a particular day. The 48 hours of RCTM data will be surrounded by missing data. The desired output file’s name is passed to the RCTM as a command line argument. A sample CDL file can be found on the public domain CD at /mdss_data/rctm/static_data/cdl/road_cond.cdl.

The CDL file describing this netCDF file format is:

```cdl
netcdf road_cond {
    dimensions:
        max_site_num = 25;  // number of locations
        num_times = 48;     // number of times road conditions are computed
        days = 4;           // number of days
```
fc_times_per_day = 24;  // fcst times per day
max_node = 150;

variables:
  int type;
  type:long_name = "cdl file type";

doubleforc_time;
  forc_time:long_name = "time of earliest forecast";
  forc_time:units = "seconds since 1970-1-1 00:00:00";

doublecreation_time;
  creation_time:long_name = "time at which forecast file was created";
  creation_time:units = "seconds since 1970-1-1 00:00:00";

intnum_sites;
  num_sites:long_name = "number of actual_sites";

intsitelist(max_site_num);
  site_list:long_name = "forecast site id numbers";

floatroad_T(max_site_num, days, fc_times_per_day);
  road_T:long_name = "road surface temperature";
  road_T:units = "degrees Celsius";

floatroad_subsurface_T(max_site_num, days, fc_times_per_day, max_node);
  road_subsurface_T:long_name = "road subsurface node temperature";
  road_subsurface_T:units = "degrees Celsius"; // first node is deepest,
    // last is surface node

floatsnow_depth(max_site_num, days, fc_times_per_day);
  snow_depth:long_name = "snow depth on road";
  snow_depth:units = "mm";

floatmobility(max_site_num, days, fc_times_per_day);
  mobility:long_name = "net mobility";
  mobility:units = "non-dimensional";

floatchemical_concentration(max_site_num, days, fc_times_per_day);
  chemical_concentration:long_name = "chemical concentration";
  chemical_concentration:units = "unknown";

floatavailable_chem(max_site_num, days, fc_times_per_day);
  available_chem:long_name = "pure de-icing chemicals on road";

floatnominal_chem(max_site_num, days, fc_times_per_day);
  nominal_chem:long_name = "theoretical chem concentration";

floatavailable_H2O(max_site_num, days, fc_times_per_day);

available_H2O:long_name = "water available for chemical dilution";

float apply_chem(max_site_num, days, fc_times_per_day);
apply_chem:long_name = "apply chemicals";

float do_plowing(max_site_num, days, fc_times_per_day);
do_plowing:long_name = "do plowing";

float treatment_time(max_site_num, days, fc_times_per_day);
treatment_time:long_name = "treatment time";
treatment_time:units = "sec since start of forecast period";

float chem_type(max_site_num, days, fc_times_per_day);
chem_type:long_name = "chemical type";
chem_type:units = "gm/cc?";

float application_rate(max_site_num, days, fc_times_per_day);
application_rate:long_name = "chemical application rate";
application_rate:units = "lb/lane mile";

float road.TempF(max_site_num, days, fc_times_per_day);
road.TempF:long_name = "road surface temperature";
road.TempF:units = "degrees Fahrenheit";

float snow_depth_inches(max_site_num, days, fc_times_per_day);
snow_depth_inches:long_name = "snow depth on road";
snow_depth_inches:units = "in";

float frost_potential(max_site_num, days, fc_times_per_day);
frost_potential:long_name = "potential for frost on road (0=low, 3=high)";
frost_potential:units = "index";

float precip_type(max_site_num, days, fc_times_per_day);
precip_type:long_name = "precip type on road";
precip_type:units = "0=NONE, 1=RAIN, 2=SNOw, 5=ICE";

data:
  type = 2;
}

13.7 Processing

The road_cond processing consists of initialization steps followed by a loop, which processes the road conditions for every site. At the end of each iteration through the site loop, the road condition data are stored in arrays formatted for writing to the output file. Once the site loop has completed, the data arrays are written to the output file.
The processing flow is described here and shown schematically in Figure 13.1.

Figure 13.1. RCTM processing for any given site.
Flow:

1. Read the site list file and set up associated data structures.
2. Read the traffic and road subsurface configuration data file.
3. Read the roadway material characteristics file and initialize the subsurface configuration for each roadway site.
4. Read in the previous `road_cond` output file (if available) and store the subsurface temperature profiles, snow_depth, and chemical attributes.
5. Read the weather forecast data file and create the weather data time series structure.
6. Create the output file and read parameters set within the output CDL file.
7. Write the forecast time, site list, etc. to the output file.
8. Allocate and initialize output data array space.

In the loop over the sites, the following operations are performed:

1. Initialize data values for this site. Extract site-specific weather, traffic, subsurface temperature profile, etc.
2. Begin loop over required treatments
   2.1 Calculate Road Temperature and Snow Depth
      2.1.1 Calculate IR Flux
      2.1.2 Create SNTHERM input files from weather and subsurface temperature data.
      2.1.3 Run SNTHERM externally, wait for its completion.
      See Appendix B.
      2.1.4 Parse the SNTHERM output file to obtain road temperature and snow depth time series.
   2.2 Calculate the mobility index at each forecast lead-time. See Appendix D.
   2.3 Determine if more treatments are required.
      2.3.1 If no treatment option is selected, no more treatments are performed.
      2.3.2 If user defined treatments are selected, read treatment list to determine next treatment (if any).
      2.3.3 If suggested treatment option is selected, the Rules of Practice module determines if another treatment is required.
      See Appendix E.
   2.4 Calculate the chemical concentration on road surface due to any treatments.
      See Appendix C.
3. If a treatment was applied, go to top of loop over required treatments to recalculate road conditions and treatments. This next pass through the loop starts at the treatment application time and overwrites existing data beyond that time.
4. Store this site’s data in arrays to be written to the output data file.

After the site loop has been completed, the following step is performed:

1. Write the output data arrays to the output file and close the file.
2. Exit with successful status.
14 DISPLAY DATA REFORMATTER

The fcst2bin source code can be found on the public domain CD in /src/apps/fcst2bin. The binary can be found at /bin/fcst2bin.

14.1 Identification

fcst2bin

14.2 Type

Process

14.3 Purpose

The purpose of the Data Reformatter is to extract and translate data into the format expected by the display.

14.4 Function

The RWFS and RCTM create netCDF files with similar formats. The Display Data Reformatter reads a file of this format, extracts the variables listed in a configuration file, and writes these variables to a file in the format desired by the display. The output may contain a time series of data (forecasts) or a single time (RWIS observations).

14.5 Dependencies

The sites listed in the site list configuration file used by this process must be a subset of the input file site list. See section 9.2 of this document for site list file specifications. The variables listed in the variable list configuration file must all exist in the input file.

14.6 Interfaces

14.6.1 Command Interface

% fcst2bin forecast_time input_file site_list var_config_file output_file [-l log_file]

where:

forecast_time is the UNIX time of the first forecast of the 48 hour forecast period. This will be truncated to the top of the hour.
input_file is the name of the file containing data to be extracted.
site_list is the name of the text file containing the list of sites to be processed.
var_config_file is the name of the configuration file containing the list of variables that will be extracted
output_file is the name of the display-ready formatted file to be produced.
log_file is the file to which log output should be written.

14.6.2 Start Script

The fcst2bin program is started by a python script, which gathers arguments and current real-time data filenames. The following list of scripts is used to do the conversions:

ep_conv_wx.py converts weather forecast data
ep_conv_meso.py converts RWIS observation data
ep_conv_no_tmt.py converts the no treatment option data
ep_conv_rec_tmt.py converts the recommended treatment option data
ep_conv_cur_tmt.py converts the current (selected) treatment option data

Each of these "wrapper" scripts sets variables (static files, etc) and then runs another script called run_proc.py which in turn executes fcst2bin using the most recent real-time data. These scripts are included with the public domain CD in the /scripts/python subdirectory.

Fcst2bin can also be invoked by the user_treatment.py script (located in the same /scripts/python directory above) via the TUNL. This script constructs the command line, runs the program, and passes the output file back to the display.

14.6.3 Configuration Files

14.6.3.1 Site List Configuration File

This file contains information on the RCTM output sites. The format of this file is described in section 9.2 of this document.

14.6.3.2 Variable List Configuration File

The variable list configuration file is a simple ASCII text file which lists each variable that the user wishes to be converted into the output file. Each line in the file consists of a single variable name exactly as it appears in the input file's CDL. These files are located on the public domain CD under /mdss_data/rctm/static_data/config.

The current contents of the weather conversion file (disp_wx_var_names.asc) are:
TempF
dewptF
rh_pct
wind_speed_mph
wind_dir
precip_rate_inches
precip_type
precip_accum_inches
snow_rate_inches
snow_accum_inches
snow_accum_total_inches
prob_precip03_pct
visibility
cprob_rain
cprob_snow
cprob_ice
blowing_snow_potential

The current contents of the road condition conversion file (disp_rc_var_names.asc) are:

road_TempF
snow_depth_inches
mobility
chemical_concentration
do_plowing
apply_chem
treatment_time
chem_type
application_rate
frost_potential
precip_type

14.7 Input

The weather forecast and road condition forecast data file are in netCDF format. This process will work with any forecast file having the same layout. That is, the same number of days and forecast times per day as in the CDL files described in sections 13.6.4.1 and 13.6.5.

14.8 Output

The reformatted data file created is in the format expected by the display. See section 16.6.2.4 for a description of that format.
15 Treatment Update Network Layer (TUNL)

The Treatment Update Network Layer is a collection of scripts that handle requests initiated by the MDSS Display. This, more often than not, involves consulting the file system and returning a particular facet of the state of the system. In the cases where the display determines that forecast data are required, the TUNL scripts gather the data, either by directly consulting the file system or by invoking processes to generate the requested data.

The TUNL scripts are on the public domain CD in the directory /scripts/cgi.

15.1 Identification

N/A

15.2 Type

Subsystem, i.e. collection of interconnected scripts.

15.3 Purpose

The MDSS display needs to interact with the MDSS data store in order to obtain the data it is tasked with presenting. To achieve this it consults the MDSS data store machine through its web server. TUNL software provides the functionality to query the state of the system. The majority of the queries are done through polling. That is, the display regularly queries the MDSS system to determine, for example, the current time and whether new data are available. If new data are available, the display can download the new data through the TUNL. The TUNL also provides the functionality to generate road conditions based on user-defined treatment plans. The python CGI scripts that may be invoked by the web server are listed below.

15.3.1 Get Current Time Script

The *time.php* script gets the system time from the web server machine. The web server’s system clock is assumed to be synchronized with the rest of the MDSS system. Use of this script ensures that the display is not affected by a host whose system clock is not properly set. The display polls to ensure that the displayed system time is current.
15.3.2 Get Latest Data Script

The *latest_data.php* script gets the latest data from the system data store. It can be called in either of two modes. It can return the valid time of the latest data or return the data itself. The former is typically used in a polling fashion to determine if new data has been generated. If it determines that new data exists, the second mode is used to download that data.

15.3.3 Get Closest Data Script

The *closest_data.php* script is similar to *latest_data.cgi*. However, rather than finding the latest data, it finds the data closest in time to the time argument passed by the display.

15.3.4 Get Permissions Script

The *get_permission.php* script checks the permissions of the requesting host. The script searches for the host name in several system files. These files contain lists of hosts that are allowed to perform certain operations. These operations include whether the display user is allowed to 1) see the current plan for the given segment, 2) select treatments for the segment, or 3) perform a user-defined treatment scenario.

15.3.5 Select Treatment Plan Script

The *select_plan.php* script saves the selected treatment plan. This selected plan contains the course of action that the garage intends to follow on the specified route. Only those users who have proper permissions can save the selected plan for a route. This is usually someone working at the DOT garage in charge of that route.

15.3.6 User Defined Treatment Plan Script

The *user_defined_plan.php* script runs a treatment plan defined by the user within the display. This involves gathering and creating a number of configuration files used by the RCTM. The process *road_cond* is invoked to run the specified plan and determine the resultant road conditions.

15.3.7 Reset Sites Script

The *resetSites.php* script allows the user to reset the road conditions within the RCTM data store for specified sites. This action may be required because the reality of the road situation does not match the system’s model of reality. For Release 3, the only allowed actions are a reset of the snow depth and chemical mass on the road to zero. That is, the
user should invoke this script when it is known that there is no snow and/or no chemicals on the road and the display indicates that the system thinks otherwise. The effect of the reset action will be seen at the forecast update which will be created starting with a clear road. Permissions are checked to determine whether the user is allowed to reset the indicated site(s).

15.3.8 Get Reset Sites Script

The *getResetSites.php* script queries the system to determine which sites are scheduled to be reset in the next forecast generation cycle. This allows the display to indicate to the users the status of that list.

15.4 Function

TUNL is a collection of CGI scripts written in Python. Each CGI script is directly invoked by the web server. The CGI scripts parse arguments and then either directly query the file system, invoke other Python scripts, or run C++ programs. Each successive invocation returns data to its parent. Eventually, the requested data is returned to the CGI script that passes it back to the display. The first Python script called also sets a timer and monitors its children to ensure that the processing has not hung. Should the processing either fail or time out, a failure message is returned to the display.

15.5 Dependencies

TUNL requires read/write access to the MDSS data directory tree which is located based on the value of the MDSS_ROOT_DIR environment variable. Typically write access can be restricted to small subtrees. These write-accessible subtrees include directories containing users’ treatment plans, temporary work space used in the generation of user-defined treatment plans, and request log files.

TUNL also requires properly formatted requests from the MDSS display. Improperly generated or communicated requests are gracefully handled.

15.6 Interfaces

15.6.1 Command Interfaces

As with all CGI scripts, the inputs are in ASCII text. All the scripts return data to the display. The returned data can either be text or binary data depending on the query. The data type is described in the first part of the returned message. The content type headers
returned to the display are either “Content-Type: text/plain” or “Content-Type:text/octet-stream”. Following these data type headers is a return string indicating the status of the request. Possible values are “SUCCESS” or “FAILURE”. The data are returned afterwards in the indicated format.

The following scripts are all CGI scripts written in the python language. They can be found on the public domain CD in the /scripts/cgi directory.

15.6.1.1  Get Current Time Script

Name: time.php

Inputs: None

Outputs:

<table>
<thead>
<tr>
<th>Data</th>
<th>Type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>Text</td>
<td>Unix time of data found (if data = “filetime”)</td>
</tr>
<tr>
<td>file</td>
<td>Binary data</td>
<td>Requested data file contents (if data = “file”)</td>
</tr>
</tbody>
</table>

15.6.1.2  Get Latest Data and Get Closest Data Scripts

Name: latest_data.php, closest_data.php

Inputs:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Definition</th>
<th>Recognized Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>Data Location</td>
<td>“nt”</td>
<td>no treatment road_cond run request</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“ct”</td>
<td>current treatment plan road_cond run request</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“st”</td>
<td>suggested treatment plan road_cond run request</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“wx”</td>
<td>weather forecast request</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“rwis”</td>
<td>weather observation request</td>
</tr>
<tr>
<td>Data Requested</td>
<td></td>
<td>“filetime”</td>
<td>Data time request</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“file”</td>
<td>Data file request</td>
</tr>
<tr>
<td>Host</td>
<td>Display Hostname</td>
<td>Used only if location = “ct”</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>time*</td>
<td>Used with closest_data.php -- not with latest_data.php</td>
</tr>
</tbody>
</table>
Output:

<table>
<thead>
<tr>
<th>Data</th>
<th>Type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>Text</td>
<td>Unix time of data found (if data = “filetime”)</td>
</tr>
<tr>
<td>file</td>
<td>Binary data</td>
<td>Requested data file contents (if data = “file”)</td>
</tr>
</tbody>
</table>

15.6.1.3 Get Permissions Script

Name: get_permission.php

Inputs:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Definition</th>
<th>Recognized Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Site ID number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>request</td>
<td>Permissions request type</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>“isSelectAllowed”</td>
<td></td>
<td>Treatment plan selection permitted?</td>
</tr>
<tr>
<td></td>
<td>“isPerformTreatmentAllowed”</td>
<td></td>
<td>User-defined treatment calculation permitted?</td>
</tr>
<tr>
<td></td>
<td>“isViewCurrentAllowed”</td>
<td></td>
<td>Viewing current treatment plan permitted?</td>
</tr>
<tr>
<td>Host</td>
<td>Display Hostname</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Output:
If the action is allowed, “SUCCESS” is returned. Otherwise, “FAILURE” is returned.

15.6.1.4 Select Treatment Plan

Name: select_plan.php

Inputs:
Note that for Release 3, only 3 chemical types are currently supported. They are NaCl, CaCl₂, and MgCl₂. The others are placeholders within the display code.

**Output:**
The script returns “SUCCESS” or “FAILRE” indicating whether or not the treatment selection was accomplished.

### 15.6.1.5 User Defined Treatment Plan

**Name:** user_defined_plan.php

**Inputs:**

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Definition</th>
<th>Recognized Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Site ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>numTx</td>
<td>Number of Treatments in Plan</td>
<td>≥ 0</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>Treatment time</td>
<td>Comma separated list of numTx UNIX times</td>
<td></td>
</tr>
<tr>
<td>rate</td>
<td>Chemical Application Rate</td>
<td>Comma separated list of numTx chemical application rates in lb/lane-mi</td>
<td></td>
</tr>
<tr>
<td>chem</td>
<td>Application type</td>
<td>Comma separated list of numTx application types</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Plow Only</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>NaCl</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>CaCl₂</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>MgCl₂</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>CaMg Acetate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>K Acetate</td>
<td></td>
</tr>
</tbody>
</table>
Note that for Release 3, only 3 chemical types are currently supported. They are NaCl, CaCl₂, and MgCl₂. The others are placeholders within the display code.

Output:
If successful, returns the binary file containing the forecast road conditions that was generated based on the specified treatment plan.

15.6.1.6 Reset Script

Name: resetSites.php

Inputs:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Definition</th>
<th>Recognized Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>Name of machine making request</td>
<td></td>
<td>Hostname needs to match name in system permissions file</td>
</tr>
<tr>
<td>site</td>
<td>Site to have road conditions reset</td>
<td>SiteID</td>
<td>Multiple sites can be reset within the same query by providing repeated site variables</td>
</tr>
</tbody>
</table>

Output:
The script returns “SUCCESS” or “FAILURE” indicating whether or not the reset was accomplished.
15.6.1.7 Get Reset Sites Script

Name: getResetSites.php

Inputs:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Definition</th>
<th>Recognized Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Name of machine making request</td>
<td></td>
<td>Hostname needs to match name in system permissions file</td>
</tr>
</tbody>
</table>

Output:

<table>
<thead>
<tr>
<th>Data</th>
<th>Type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>sites</td>
<td>Text</td>
<td>List of sites scheduled to be reset</td>
</tr>
</tbody>
</table>

16 DISPLAY

The MDSS Demo Display is a Java application that retrieves data from the server modules RWFS and RCTM, and displays them for the user. A tutorial of the display (PowerPoint) from the winter 2004 demonstration can be found at:

http://www.rap.ucar.edu/projects/rdwx_mdss/MDSS_demo_tutorial_v301_Jan04.pps

The tutorial provides an easy reference for the major features and functions of the display application.

The architectural goal for the display was to keep it as thin as possible, pushing decision-making algorithms into the server modules as much as possible. Some complexity exists due to the requirement that the application hide information from some users who have a restricted viewing permission level, but this is handled for the most part by the TUNL layer, described above. The other area of complexity lies in the calculation of alerts. The details of this are described in this section.

The Java archive files, MDSS.jar and jdom.jar, can be found on the public domain CD in /src/apps/display/edu/ucar/rap/mdss/apps/iowa.

Note, when downloading Java WebStart, the display application and running the live MDSS display, web filtering software (e.g. WebNanny, Cybersitter) should be temporarily disabled.

16.1 Identification

MDSS.jar
16.2 Type

Java Process

16.3 Purpose

The purpose of the display system is to demonstrate the functionality of the various MDSS components by providing a graphical, interactive view of the output data. In this way the display system can be a starting point for discussions of functional requirements for operational systems based on or incorporating components of MDSS. The 2004 MDSS demo display features and functions were developed with feedback from representatives from IA, MN, WA, UT, and NH DOTs. The domain and routes used in the 2004 system were located in the Des Moines, IA area.

16.4 Function

The display system reads output data from the Display Data Reformatter and provides the user with various derived views of the data. It also provides the user with an interface to enter user-specified treatments. The treatments are submitted to the RCTM to predict the resulting effect on road condition.

The 2004 prototype MDSS display system contains a map that provides two levels of zoom – a “state view” and two different “route views.” At the state view, weather information and current road temperature observations are available on the map, but not road condition information. Within the route views, both weather and road condition information are available on the map. For the Iowa demonstration, weather information was provided for 119 forecast points within the state, and road condition information was provided for 16 road segments.

The display system provides high-level alerts of forecasts for inclement weather, impaired road conditions, and blowing snow. Weather alerts are provided by grouping the weather conditions into distinct weather alert categories, as shown in Table 16.1 The weather alerts are provided in four ways: 1) A state map shows the worst weather alert category that will occur over the forecast period for each forecast region (forecast regions were defined by IADOT), 2) A weather alert time bar shows the worst weather alert category for the currently-displayed portion of the state during three forecast time ranges: 0-12 hours, 12-24 hours, and 24-48 hours, 3) Cursor-over inspection on a point-by-point basis shows a color bar indicating the weather alert category for each point at every forecast hour, when any variable other than road temperature observations is selected, and 4) Colored dots show the weather alert category of each forecast point at the selected time when the weather alert category forecast product is selected.
Weather Alert Category Table (Release-3 version)

<table>
<thead>
<tr>
<th>Weather Alert Category</th>
<th>Precip Type</th>
<th>Precip Rate (Liquid Equiv.) (inches/hr)</th>
<th>Temperature (F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extreme</td>
<td>Any ice</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Snow</td>
<td>&gt;= 0.15 “/hr</td>
<td></td>
</tr>
<tr>
<td>Poor</td>
<td>Snow</td>
<td>0.05-0.15 “/hr</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rain</td>
<td>&gt;= 0.25 “/hr &lt; 35 F</td>
<td></td>
</tr>
<tr>
<td>Marginal</td>
<td>Snow</td>
<td>Any</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rain</td>
<td>&gt;= 0.1 “/hr</td>
<td></td>
</tr>
<tr>
<td>OK</td>
<td>All other conditions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 16.1. Weather conditions/alert category table. A weather alert level is derived from the forecast weather conditions. This table describes the groupings of weather conditions with weather alerts.

Road condition alerts are provided by grouping road conditions into four distinct road condition alert categories as shown in Table 16.2. In this case, road condition alert categories are based entirely on the computed mobility index. Road condition alert categories are displayed in three ways: 1) A road condition alert time bar shows the worst road condition alert category for the currently-displayed portion of the state during three forecast time ranges: 0-12 hours, 12-24 hours, and 24-48 hours, 2) Cursor-over inspection on a point-by-point basis shows a color bar indicating the road condition alert category for each road segment at every forecast hour, and 3) Colored road segments show the road condition alert category for the selected time when the user is zoomed in to one of the “route views” in the display. All road condition alerts are based on the predicted road condition after the user-selected treatment has been applied.

Road Condition Alert Category Table

<table>
<thead>
<tr>
<th>Road Alert Category</th>
<th>Mobility Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extreme</td>
<td>0 – 0.25</td>
</tr>
<tr>
<td>Poor</td>
<td>0.26 – 0.50</td>
</tr>
<tr>
<td>Marginal</td>
<td>0.51 – 0.75</td>
</tr>
<tr>
<td>OK</td>
<td>0.76 – 1.0</td>
</tr>
</tbody>
</table>

Table 16.2. Road conditions/alert category table. A road alert level is derived from the net mobility values. This table describes the mappings of net mobility values to road alerts.
Blowing snow alerts are generated for each weather forecast site, for each forecast hour. The algorithm for calculating the blowing snow alert categories is not in the display; alerts are received from the TUNL and are shown without any re-interpretation. The alerts fall into the same four categories used by the weather alerts and road alerts: OK, Marginal, Poor, and Extreme. Blowing snow alerts are shown in 3 ways: 1) a blowing snow alert time bar shows the worst blowing snow alert category for the currently-displayed portion of the state during three forecast time ranges: 0-12 hours, 12-24 hours, and 24-48 hours, 2) a blowing snow alert time bar shows the blowing snow category for each hour for the currently selected route when the display is zoomed into one of the route views, and 3) the same blowing snow alert time bar is shown in the event summary dialog of each route.

Road frost alerts have been identified as a potential future alert category (development planned for 2005). Though data are not currently being generated in MDSS, the display has been designed to show them. Locations in which to show road frost alerts have been set aside as: 1) a road frost alert time bar showing the worst blowing snow alert category for the currently-displayed portion of the state during three forecast time ranges: 0-12 hours, 12-24 hours, and 24-48 hours and 2) a road frost alert time bar showing the blowing snow category for each hour for the currently selected route when the display is zoomed into one of the route views. Both of these time bars are currently grayed out to indicate the lack of data.

Road treatment functionality is provided in the Treatment Selector dialog. This dialog allows the user to get a recommended treatment (based on Rules of Practice) for each of the road segments modeled in the MDSS Functional Prototype. It also allows the user to test user-specified treatments for each segment. Once the user is satisfied with the results of a treatment scenario, the user can use the Treatment Selector dialog to select one of the scenarios to use in operations, and the display adjusts the road condition alerts to reflect the predicted road conditions based on the selected treatment.

A summary of predicted weather and road conditions is provided in the Event Summary dialog. There is an Event Summary dialog available for each route in the system. The dialog shows the relative probabilities of rain, snow, and ice, as well as the total probability of precipitation and declared precipitation type, if any. The dialog also shows total snow accumulation without plowing, road temperatures with recommended treatment, and wind speeds for each hour, as well as 48hr maximums and minimums. Finally, the dialog provides blowing snow alert categories and the system's treatment recommendation. The dialog contents may be printed.

The Treatment History dialog for each route displays the recommended and last-selected treatments for the current 3-hour run and the two previous 3-hour runs. The information is presented in graphical or textual format, as desired. The dialog contents may be printed.

The Configuration dialog provides access to available system configuration options. After a storm has completed, it is possible that the MDSS's assumed current road conditions have drifted from reality. In this case, it may be desirable to reset the system's initial conditions to a state consistent with a clear road and no residual chemical concentration. The “Reset
Segments” portion of the configuration dialog allows routes to be reset individually or as a group. The reset routes' initial conditions are reset for the next 3-hour system run. The Configuration dialog also allows the default shift times to be changed for all routes. At this time, the shift times do not affect the recommended treatments or any part of the RCTM. They are only shown on the graphical treatment bars as a scheduling convenience.

16.5 Dependencies

The display system is contained entirely in two Java archive (jar) files. These java archives contain the Java byte code necessary to run the system. The file ‘MDSS.jar’ contains the byte code, some GIF images loaded by the display to provide graphic icons for ease of use, and several configuration files. The file ‘jdom.jar’ contains APIs (Application Programming Interface) for parsing XML configuration files. jdom.jar is freely available from the JDOM Project at http://www.jdom.org. It is used in compliance with the license file provided. There are no dependencies outside these two .jar files for the demonstration. These .jar files can be found on the public domain CD in /src/apps/display/edu/ucar/rap/mdss/apps/iowa.

16.5.1 Runtime Environment

The display system can be run on any system, which has Java 1.4.x installed. Platforms on which the system have been tested include Solaris, Solaris for Intel, Debian Linux 2.1, Debian Linux 2.2, RedHat Linux 7.3, Windows NT, Windows 2000, and Windows XP.

The Java 1.4.x runtime environment must be installed on the hardware running the demonstration display application. To download and install this software package, see the Sun Microsystems website at: http://java.com. Click on the link to “Free Download: Java software for the desktop.” The current version is Java 1.4.2. It includes a distribution of Java WebStart.

The user running the display demonstration must include the path to the JDK1.4.x bin directory in his or her path. See your system documentation for information about setting the path for your particular platform.

16.5.2 Java WebStart Deployment

The display system must be deployed to other clients via a web server using Java WebStart, available from Sun Microsystems. WebStart is installed on the client machines of users downloading the display system as described above. The only change necessary to the web server deploying the application is to make an entry in the configuration file mapping a new MIME type to the .jnlp file extension used for deploying applications in this way. For the Apache web server, the following entry is used in the file /etc/apache/httpd.conf:
Instructions for downloading and installing Java WebStart are available at the Sun Microsystems web site: http://java.sun.com/products/javawebstart/. For all required 1.4.x Java versions, JavaWebStart is bundled with the virtual machine, so no additional download is necessary. It is possible to streamline the setup process for the user by providing a web page on the server that uses javascript to test whether WebStart is installed, and directing the user to the appropriate URL to download the necessary WebStart application. An example of such a script is available at: http://www.rap.ucar.edu/projects/rdwx_mdss/demo.html. To view this example, browse to this location, then select “View Page Source” in your browser to see the JavaScript code.

The web server, where the Java archives are placed for access by Java WebStart clients, must contain an xml document showing the configuration of the MDSS Display Application. This document, known as a “.jnlp file” (Java Network Launching Protocol) tells Java WebStart how to invoke the Display Application. Users with Java WebStart properly installed will have Java WebStart automatically invoked by their browsers when they surf to the .jnlp file. Note, when downloading Java WebStart, the display application and running the live MDSS display, web filtering software (e.g. WebNanny, Cybersitter) should be temporarily disabled. Web filtering software corrupts the authentication encryption of the application's .jar files.

Below is an example .jnlp file, used to deploy the live demo application in the winter 2004 season.

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-
A JNLP File for MDSS Demo Application.
@version 2.0
@date 2003/10/31
@author Arnaud Dumont
-->

<jnlp spec="1.0+">
  codebase=http://www.rap.ucar.edu/projects/rdwx_mdss/demoLive
  href="mdss.jnlp">

  <information>
    <title>Iowa MDSS Live Display</title>
    <vendor>NCAR - RAP, FHWA - RWMP</vendor>
    <homepage href="http://www.rap.ucar.edu/projects/rdwx_mdss"/>
    <description kind="tooltip">Iowa MDSS Live Display</description>
    <description kind="one-line">Iowa MDSS Live Display</description>
    <icon href="iowaicon.gif"/>
    <icon kind="splash" href="iowasplash.gif"/>
```
16.6 Interfaces

16.6.1 Command Interface

The display system can no longer be run manually from the command line. The application must be deployed with Java WebStart. This ensures that clients have the proper runtime environment in place to run the application successfully.

When the application is deployed using Java WebStart, the application can be restarted by simply clicking on the same link that originally downloaded the display system. Java WebStart will check whether the user’s version of the display system is up to date, and will download a new version if necessary. No download will take place if the user’s version is up to date.

If the application was downloaded using Java WebStart it may be restarted from the desktop by starting up the WebStart application, then opening the MDSS Application from within the WebStart window. To start Java WebStart manually, type:

```
<path to root of Java installation>/jre/javaws/javaws
```

16.6.2 Input

The MDSS Display system reads several binary data files to determine the weather forecast and road condition forecasts. All of these files have the same format, as described below. These files are created by the Display Data Reformatter.

A binary format was chosen because it is much faster for Java to read binary data than to read ASCII data and convert it to numerical types. A binary format allows approximately 10-fold decrease in the time it takes to ingest a data file. The binary files are also smaller than their ASCII representations would be, reducing network transmission times.
16.6.2.1 Weather Forecast Data File

A single file contains all the weather forecast data from a given forecast run for the display. The file is retrieved from the data server via an apache web server. An example of such a file on the server is:

```
rcrtm/conv_wx/20030224/conv_wx.20030224.0017.bin
```

16.6.2.2 RWIS Observation Files

A single file contains RWIS observations for the latest time for the display. The file is retrieved from the data server via a web server. An example of such a file on the server is:

```
rctm/conv_meso/20030224/conv_meso.20030224.0017.bin
```

16.6.2.3 Road Condition Forecast Files

Three treatment type data files contain the standard road condition forecast data – one for the road condition with no treatment, one for the road condition with the recommended treatment, and one for the current (selected) treatment. Three recommended and selected treatment files and one no-treatment file are retrieved from the data server via an apache web server. The three recommended and selected treatment files are from the most recent model run and the two previous runs. The no treatment file is from the most recent model run. Examples of such files on the server are:

```
rctm/conv_no_tmt/20020224/conv_no_tmt.20030224.0019.bin - No Treat.
rctm/conv_rec_tmt/20020224/conv_rec_tmt.20030224.0019.bin – Rec. Treat.
rctm/conv_cur_tmt/20020224/conv_cur_tmt.20030224.0019.bin – Cur. Treat
```

16.6.2.4 User-Specified Treatment Road Condition Forecast Files

Files are provided to the display system when user-specified treatments are submitted to the RCTM module via remote communications. These files contain the solution to such ‘what-if’ scenarios. These files are named based on the following convention:

```
rctm/user_treatment/<host>/<site_id>/conv_rc_ut/<YYYYMMDD>/conv_rc_ut.<YYYYMMDD>.<HHMMSS>.bin
```

Where:

- `<host>` is the hostname of the client requesting the user treatment
- `<YYYYMMDD>` are the Year, Month, and Day of the forecast start, in UTC
- `<HHMMSS>` is the hour, minute, and seconds of the data (UTC)
<site_id> is the location number of the forecast site, currently one of:

74449036 - IA MDSS Seg 1
74449037 - IA MDSS Seg 2
74449038 - IA MDSS Seg 3
74449039 - IA MDSS Seg 4
74449040 - IA MDSS Seg 5
74449041 - IA MDSS Seg 6
74449042 - IA MDSS Seg 7
74449043 - IA MDSS Seg 8
74449044 - IA MDSS Seg 9
74449045 - IA MDSS Seg 10
74449046 - IA MDSS Seg 11
74449047 - IA MDSS Seg 12
74449048 - IA MDSS Seg 13
74449049 - IA MDSS Seg 14
74449050 - IA MDSS Seg 15
74449051 - IA MDSS Seg 3B

16.6.2.5 Input File Format

All the weather and road condition files contain the same binary format. The format is as follows:

The following sequence is repeated for each forecast station:

- Int station_id
  The following sequence is repeated for each variable name for the station:
  - Char[40] variable_name
  - Int num_days
  - Int vals_per_day
  - Double base_time
  - Float[num_days * vals_per_day] values
  - Char[40] "#                                       " – indicates end of a station’s data

Where:
- Station_id is a numerical station identifier
- Variable_name is one of:
  "TempF"
  "dewptF"
  "rh_pct"
  "wind_speed_mph"
  "wind_dir"
  "precip_rate_inches"
  "snow_rate_inches"
"precip_type"
"prob_precip03_pct"
"cprob_rain"
"cprob_snow"
"cprob_ice"
"precip_accum_inches"
"snow_accum_inches"
"snow_accum_total_inches"
"visibility"
"alert_cat"
"blowing_snow_potential"
"road_TempF"
"snow_depth_inches"
"mobility"
"chemical_concentration"
"do_plowing"
"apply_chem"
"chem_type"
"treatment_time"
"application_rate"
"frost_potential"
"roadTemperature"
"roadTemperature1"
"roadTemperature2"
"roadTemperature3"
"roadTemperature4"

num_days is the number of full days in the forecast
vals_per_day is the number of forecast values provided per day
base_time is the UNIX time of the first data value, in UTC time
values is the array of data values for the variable

16.7 Output

The display system does not generate any output files. Debug information is available from within the Java WebStart Java console. To enable this output from within WebStart, start WebStart as described above, and select the menu item “File”->”Preferences.” A dialog will appear, in which you should select the “Advanced” tab, and click the “Show Java Console” checkbox.

17 Future MDSS Enhancements and Refinements

The MDSS prototype will continue to be enhanced and refined in FY2005, but the amount of development will be less than in previous years as the funding will decrease and the focus will be on technology transfer. The MDSS test bed will move from Iowa to Colorado
and another live field demonstration will be conducted in Colorado during the winter of 2004-2005. Improvements to the system will be included as part of Release-4.x, which is scheduled for September 2005.

Candidate improvements to the MDSS that will be part of Release-4.x include, but are not limited to, the following:

- A nodal parameterization tool will be developed and made available for configuring layer structure files for the SNTHERM pavement model.
- Enhancements to the Display Application, including:
  - Activation of the road frost algorithm display product
  - The provision of information on the reasons behind the treatment recommendations
  - Activation of the MgCl₂ selection
- Development of frost prediction algorithms within the RCTM
- Modifications made to the prototype during the field demonstration of winter 2005 will also be provided in Release-4.x
- Additional refinements and tested Rules of Practice
- Improvements to the RWFS to handle routes in complex terrain

Note: Although the MDSS Release-4.x system will contain more refined components, it will still be a prototype system and caution should be exercised when utilizing its components.
APPENDICES
Appendix A: Road Weather Forecast (RWFS) Subsystem

A.1. RWFS Overview

The RWFS is tasked with ingesting meteorological data (observations, models, statistical data, climate data, etc.) and producing meteorological forecasts at user defined forecast sites and forecast lead times. The forecast variables output by the RWFS are used by the RCTM to calculate the road surface temperature and to determine a suggested treatment plan. In order to achieve this goal, the RWFS generates independent forecasts from each of the data sources using a variety of forecasting techniques. A single consensus forecast from the set of individual forecasts is generated at each user defined forecast site based on a processing method that takes into account the recent skill of each forecast module.

A.2. RWFS Core Forecasts

The RWFS generates point forecasts for locations along the highway system (and elsewhere as configured). However, very few of these sites are observational sites that regularly make automatic reports. At observational sites, forecast parameter tuning based on past performance helps improve the forecasts. This class of sites is called core forecast sites. Forecasts at non-core sites are derived from forecasts at core sites.

The RWFS makes many of its forecasts based on numerical weather prediction (NWP) model data. The temporal resolution of this data determines the forecast resolution. The lead times at which model data is directly available are called core forecast times. These generally determine the temporal resolution of the computations. Forecasts at non-core forecast times are made by interpolation from core forecast times.

The variables output by the RWFS are used by the RCTM and the display. The observational data ingested only contains a subset of the variables required by these downstream processes. The RWFS output variables contained in the observational data sets are called the core forecast variables.

A.3. RWFS Forecast Subsystems

The RWFS was redesigned for the winter 2003-2004 to make use of the FSL time-lagged model ensemble data, which were provided at hourly temporal resolution out to 15 hours. This is finer scale data than the three-hourly data provided by NCEP. Effectively two versions of the RWFS were run in parallel. The first was run at higher core temporal resolution (1-hourly) out through the time extent of the FSL models. The second ran at 3-hourly temporal resolution through the end of the 48 hour period. Intermediate (hourly) forecasts for the second subsystem were generated by interpolation.

The FSL time-lagged ensemble model data is provided at one hourly resolution. This allows “core” forecast lead times at one hour intervals for the first 12 hours. The NCEP
model data is also used during this time range. The NCEP model data is interpolated to one hourly intervals for use within this subsystem, called the Near Term system.

Beyond the first 12 hours, the only available model data is from NCEP and has 3 hour resolution. So, in this time range, core forecast times are at 3 hour intervals. Forecasts at non-core forecast times are made by interpolation from core forecasts. This subsystem is called the Short Term system.

The output of these two subsystems is merged at the end of the forecast process. The first 12 hours of data are taken from the Near Term system and the remaining 36 hours are taken from the Short Term system.

The forecast module and forecast integration sections below describe features that appear in both the Near Term and Short Term systems.

A.4. RWFS Forecast Modules

The RWFS creates several independent forecast estimates. Each forecast module attempts to create the best forecast it can by applying a specific forecast technique to its input data set. Each RWFS forecast module uses one of three basic techniques to generate forecasts. They are:

- Dynamic Model Output Statistics (DMOS),
- Interpolation of NWS MOS site forecasts, and
- Semi-static techniques.

Each forecast module produces an identically formatted output file. No forecast module is dependent on another forecast module. That is, no forecast module's output is used as input to another forecast module.

Dynamic MOS forecast modules

The Dynamic MOS (DMOS) forecast modules are a dynamic variation of the traditional NWS MOS procedures. DMOS, like traditional MOS, finds relationships between model output data and observations using linear regression methods. However, while MOS equations are calculated using many years of data, DMOS uses only the last 3 months (configurable) of data. New regression equations are re-calculated once per week.

The DMOS technique has several advantages over traditional MOS. The reliance on only a short history allows DMOS equations to be calculated and DMOS forecasts generated for newly ingested models or models that are changing due to enhancements. Traditional MOS equation generation would require the model to be stable (no changes) for several years. Also, the MOS equations are calculated painstakingly with a large human quality control effort. This makes it difficult to add MOS equations for a new set of forecast sites. DMOS
forecasts can be made at these sites immediately provided they have a high quality observational history of at least three months (configurable).

A disadvantage of DMOS is that the equations it produces can be less stable than traditional MOS equations. For this reason, quality control checks must be put into place to assure that the equations produced will not create nonsensical outlier forecasts.

The DMOS subsystem applied to any model has three components:

- Regressor calculation,
- Empirical Relationships Generator, and
- Forecast Generator.

The interaction of these three components is diagrammed in Figure A1.

Figure A1: The interaction of the 3 modules of the RWFS DMOS forecast subsystem is shown. The Regressor Calculation Module extracts and derives forecast point specific data from the model data. A history of these regressor files is accumulated and the Empirical Relationships Generator finds empirical relationships between the regressors and the observed data. These empirical relationships are applied to the current day’s regressor data to generate the DMOS forecast.

A.4.1 Regressor Calculation

Regressors are variables extracted or derived from model data, which is likely to have a relationship to one of the output forecast variables. These regressors are calculated at each forecast site for each forecast lead-time. About 2/3 of the regressors are variables directly
extracted from the model data. Other regressors are derived by combining several variables to estimate meteorological data not explicitly predicted by the models.

Since the forecast sites are rarely at model grid points, interpolation techniques are used to generate forecasts at the forecast sites. This requires an understanding of the projection of the model grid and the terrain assumptions used in each model. As some of the regressors are estimates of meteorological variables at the earth's surface, correcting for the simplified terrain used by the model is important and varies from model to model.

The regressors from one model run are all stored in one file. The regressor files are put into a regressor history that the DMOS empirics process uses to calculate regression equations. DMOS Empirical Relationships Generator

The DMOS Empirical Relationships Generator attempts to find relationships between the regressors and the observations at forecast sites. It does this using a linear regression technique. There are tradeoffs involved in determining the best regression equation. The goodness of fit measure of a regression equation is called its r-squared value. Typically, adding more regressors to an equation increases the r-squared value. However, this also increases the variance of the output forecasts since more regressors are included that do not have a strong relationship to the predictand. Therefore, the desired set of regressors has most of the information leading to a good prediction and does not contain noisy regressors.

Equations that do not have a sufficiently high r-squared value are replaced with a default equation. This default equation is a predefined combination of regressors defined by a meteorologist. A default equation is an attempt to generically replicate a meteorologist's logic in coming up with a forecast. Special, usually derived regressors have been developed for this specific purpose. These default equations generally do not produce the erroneous forecasts that a low r-squared equation might.

This best combination of regressors will vary from site to site, between forecast lead times, and clearly will be different for each forecast variable. The relationships will also vary from season to season and from model to model. The empirics generator is run once per week for each model to find the equations which best fit the most recent data. These equations are stored in a DMOS empirics file and used later by the DMOS forecast generator.

**A.4.2 DMOS Forecast Generator**

The DMOS Forecast Generator applies the empirical relationships generated by the DMOS Empirical Relationships Generator to the most recent regressors. This generates the DMOS forecast. The relationships between regressors that have done well at predicting the observations recently are used again on today's regressor data to make a DMOS forecast. If any of the regressors that appear in a regression equation are missing, a missing forecast is generated.
A.4.3 NWS MOS Forecast Modules

These forecast modules are based on the MOS products generated by the National Weather Service. These forecasts are not a perfect match to the desired MDSS forecasts. The MOS data consist of point forecasts at sites chosen by the NWS. These MOS sites are generally a subset of the MDSS forecast sites. Also, the variables forecast in the MOS output varies for each of the NWS models. In addition, the variables do not directly match the MDSS forecast variables and it is possible that the forecast lead times do not match the MDSS forecast lead times.

At a site included in any particular NWS MOS forecast, the forecast module tries to reproduce the exact forecast. Where MDSS variables are explicitly forecast in the MOS product, they are simply copied. Otherwise, if reasonable, the MDSS forecast variable is derived from the MOS data. For some variables, no derivation is reasonable and these variables are left as missing data. If the forecast lead times of the MOS product do not match the MDSS forecast times, the forecast module makes an interpolated forecast where possible.

For the majority of the MDSS sites, no MOS forecasts exist. Forecasts for these sites are generated by interpolation techniques. The interpolated forecasts are generated using the forecasts generated at the MOS sites. No satisfactory interpolation technique has been found that works well for all variables in rough terrain. For example, the interpolation of surface winds in the mountains does not work well using any known technique.

A.4.4 Semi-static Forecast Modules

One forecast module, the climatology forecast module, is called semi-static in that its forecasts depend only on historical data, not on any predictive forecast model. The climatology forecast module uses data from up to the last 30 years. Monthly averages of the MDSS forecast variables have been computed and stored in a climatology file. These monthly climatological values are assumed to be valid mid-month and are interpolated to the forecast date. The climatology forecast module has more effect on the forecasts for longer-term forecast periods (> 72 hours). This module will not provide a significant contribution in the MDSS, which will be configured to only provide guidance out to 48 hours.

A.5. Forecast Integration

A.5.1 Integration Overview

The RWFS forecast modules each generate as complete a forecast as possible. This includes a forecast for every forecast variable at every forecast site for every forecast lead time. These independent forecast estimates are combined by the integrator to generate one final consensus forecast. Numerous combination techniques have been developed. Investigation has led to a decision to use an enhanced Widrow-Hoff learning method. This
method creates its final forecast using a weighted average of the individual module forecasts. The weights are modified daily by nudging the weights in the gradient direction of the error in weight space. The effect of this is that forecast modules that have been performing well for a particular forecast (variable, site, and lead time) get more weight and the poorly performing modules get less weight. Note that different weight vectors exist for every forecast generation time due to differing latencies in the input data sets. The interaction of components of the integrator is diagrammed in Figure A2.

Figure A2: The RWFS Integrator Subsystem is shown. The Integrator Empirics module generates new weights by updating the existing weights based on the performance of the forecast modules. These weights are applied to the current forecast modules output.

**A.5.2 Integrator empirics**

This RWFS process runs once per day and updates all the weights based on the performance of the various forecast modules. It reads the observations from the previous day and compares the forecast modules' output that predicted those observations. For each forecast, the errors are computed and the gradient vector in weight space is computed. A step proportional to the size of the combined error is taken in that gradient direction to compute the new weights.
A.5.3 Integrator

The integrator creates a final forecast by making a bias-corrected confidence-weighted sum of the individual module forecasts. It reads the forecasts from the forecast module output files, the weights from the integrator empirics file, performs its calculations, and stores its results.

A.5.4 Non-verifiable Data Extractor

The RWFS forecasting techniques described above only apply to core forecast variables. These are variables that are regularly measured and reported in meteorological observation data. The DMOS forecast modules and the integrator both require specific observations to tune themselves.

Some variables required by the RCTM are not included in standard meteorological observation reports. However, several of these variables are generated by numerical weather prediction models. The RWFS attempts to provide the RCTM with reasonable estimates of these variables by using a combination of various models' data. The weights used in the combination are pre-determined by a meteorologist familiar with the models and stored in a configuration file. The model variables to be combined have been extracted by the DMOS regressor calculation process and stored in a regressor file. The Non-Verifiable Data (NVD) extractor reads in the appropriate models' regressor files along with the weight configuration file before creating its weighted combination output.

A.5.5 Post Processor

The post processor provides a variety of processing options to merge the integrator's forecasts and the NVD forecasts. It attempts also to remove ridiculous forecasts, derive other forecast variables, and spatially and temporally interpolate the forecasts to non-core forecast sites. The output of the post-processor is the output that will be fed into the RCTM.

Quality control measures are applied to the integrator's output to ensure that no forecasts are well beyond reasonable ranges. Forecast values near the limits are returned to the bounding values. For example, forecasts of 101% probability of precipitation are turned into forecasts of 100%. Forecasts well beyond the bounds are replaced with a missing data flag.

The MDSS provides forecasts for numerous non-core forecast sites along the highways maintained by the DOTs. The output of the integrator contains only forecasts for core forecast sites. Forecasts at the non-core sites are generated by spatial interpolation from the core sites' forecasts.
Since the MDSS provides forecasts at one hour intervals and the core forecast lead times are at three-hour intervals beyond 12 hours, temporal interpolation of the three-hourly forecasts is used to generate the desired final forecast temporal resolution.

A Forward Error Correction (FEC) process is applied to nudge the forecasts closer to the observations. The difference between the observation and the forecast for the current time is found. That difference is used to calculate a correction to the forecasts. For each of the first forecasts, the initial forecast error is multiplied by lead-time dependent factor. This correction factor is applied to the forecast time series at each lead time.

Forecast variables required by the RCTM or the display system are derived from the core set of MDSS forecast variables. For example, relative humidity is derived from temperature and dew point temperature. Blowing Snow Potential is computed here as a derived variable by examining the observational and forecast time series.

A.6. Road Weather Forecast System – Development Status

Development Status: The core technology used within the Road Weather Forecast System (RWFS) is known as DICAST™. It was developed as an intelligent data fusion system for public (domestic and international) forecasting applications. An operational version of the system has been implemented in the private sector and many of its methods and techniques are now being incorporated into commercial weather forecasting systems. The RWFS was reapplied for the MDSS application. The RWFS, (as an intelligent data fusion capability) is quite mature. Development is ongoing, but the current version is very robust and routinely produces better forecasts than individual inputs.
Appendix B: Road Temperature Module

The Road Temperature Module, SNTHERM-RT, [Jordan, R. (1991) A One-Dimensional Temperature Model for a Snow Cover Technical Documentation for SNTHERM.89, CRREL Special Report 91-16] is a land-surface model developed at Cold Regions Research Engineering Laboratory (CRREL) that generates the road surface temperature and subsurface temperature profile. This snow/road/soil temperature model is a one-dimensional mass and energy balanced model constrained by meteorological boundary conditions. The model considers the transport of liquid water and water vapor, and phase changes of water (except in the road layer) as components of the heat balance equation. The impact of snow accumulation, ablation, densification, and metamorphosis on the snow thermal and optical properties are modeled. The infiltration of water in the snow/soil is modeled assuming gravity flow.

The model presently does not consider the flow of water in frozen soil or in the road layer (asphalt or concrete). To accurately represent the flow of water in soils, the capillary pressure effects must be modeled. When snow is present, the water infiltrating to the snow/road surface is artificially drained from the system. The snow, road, and soil are divided into horizontally infinite control volumes (referred as nodes) and the mass and heat balance equations are applied to each control volume. A spatial discretization scheme similar to a finite-difference method is used in the spatial domain, while a Crank-Nicolson method is used to discretize the time domain. The model uses an adaptive time step procedure that adjusts the time step (typically between 5 and 900 seconds) to obtain the desired accuracy of the solution to the mass and heat balance equations. The governing functions are linearized and a tri-diagonal matrix algorithm is used to obtain the desired solution.

The physical, thermal, and optical properties of the horizontally infinite layers must be specified. The number of layers and the thickness of the layers is part of the physical properties that must be specified. The lowest layer in the model is set to a constant temperature. The flexibility to specify the layer properties can be used to model an overpass or a bridge. This is accomplished by using a sufficient number of layers to define the overpass structure plus one additional layer that represents the air under the bridge. The temperature of this lowest layer is set to the ambient (air) temperature. Preliminary model runs indicate the bridge temperatures differ from the surrounding road temperature in a physically consistent fashion.

SNTHERM was developed in the 1980s in FORTRAN as a research model. SNTHERM's I/O is completely file-based. It reads three input files and produces one output file. The output file contains the temperature profile and the snow depth at each forecast lead-time.

SNTHERM was prepared for single run applications and not for repeated iterative use on a number of sites. Significant re-workings of the code would have been required to achieve usability as a subroutine called from within the RCTM. For this reason, SNTHERM is run as a separate process invoked by the RCTM. The RCTM driver creates the input files and
starts SNTHERM. When SNTHERM has completed its run, control is passed back to the RCTM driver that parses the SNTHERM output file.

The input files passed to SNTHERM includes one master file FILENAME that contains the file names of the other input files and the output file. One of the other input files contains the meteorological predictions while a second file contains the SNTHERM configuration information and the road subsurface structure and initial conditions.

For detailed technical information on SNTHERM, the reader is directed to CRREL (see appendix I).

**SNTHERM References**


Appendix C: Chemical Concentration Module

The chemical concentration code is part of the Road Condition and Treatment Module. The code can be found on the public domain CD in the directory /src/apps/road_cond. The file of interest is ChemConc.cc.

C.1. Algorithm Identifier

ChemConc

C.2. Type

Subprocess

C.3. Software

ChemConc.cc

RulesOfPractice.hh

C.4. Conceptual Overview

The ChemConc algorithm is designed to estimate the chemical concentration of anti-icing and de-icing chemicals as they are applied during the course of a winter storm. There are a wide variety of anti-icing chemicals available to operators; this algorithm currently supports: salt (NaCl), calcium chloride (CaCl₂), and magnesium chloride (MgCl₂). Maintenance operators choose different types and forms (dry, wet, and combinations) of anti-icing chemicals based, in part, on how well the characteristics of a particular chemical match the forecasted weather conditions.

An essential characteristic of anti-icing chemicals is their ability to reduce the freezing point of water. The phase diagram shown in Figure C1 illustrates the freezing point depression characteristics for various concentrations of salt solution. At a solution concentration of 23.3% the freezing point of water is reduced to –6.02 degrees F. This point represents the peak freezing point depression for this chemical and is called the Eutectic Point. Solution and temperature combination below the bounding curve on the left will result in ice formation; the curve represents the chemical’s solution point. Conversely, solution and temperature combinations that fall to the right of bounding curve on the far side of the diagram will result in unabsorbed chemical. This curve is called the saturation curve. Ideally, anti-icing practices attempt to maintain the chemical concentrations between the solution point (no ice) and the saturation point (no wasted chemical).
In addition, practical studies have shown that the true range of anti-icing chemical effectiveness is bounded by the 50% level (½ point) of the eutectic concentration. For NaCl, the ½ point is 11.65% corresponding to a temperature of −8.0 deg C. The algorithm takes this ½ point into account (C_NACL_HALFPOINT_TEMP) when calculating the length of time chemicals will be effective. Finally, there is a parameter (P CHEM SAFE DELTA) that may be used to adjust the sensitivity of the algorithm by adding a temperature buffer above or below the eutectic curve. Positive values will make the system recommend higher treatments, while negative values will recommend lower treatments.

Figure C2 illustrates the life history of anti-icing chemicals as they are applied before, during and after a storm. Spreader trucks (or tankers in the case of liquid chemicals) are used to spread the selected anti-icing chemical. As the truck delivers the chemical, the force of the compound hitting the road causes some of the chemical to fall off the road (road splatter). Additionally, winds may blow the chemical off the pavement before it has had a chance to stick to the road. Once the chemical is applied, routine road traffic may also scatter the chemical off the roadway (traffic splatter).
As precipitation begins to occur, the chemical mixes with the available surface water to form a chemical solution. Some of the solution is lost as the liquid drains from the roadway. The strength of this solution is directly calculated from the amount of chemicals dropped and the precipitation that falls on the road surface. The anti-icing and de-icing effectiveness of the solution is determined by knowing the concentration of the solution and the temperature of the solution (pavement temperature).

As more precipitation falls, the chemical concentration continues to decrease. In addition, even without additional precipitation, the solution will slowly evaporate from or drain off the road surface. Eventually, the chemical concentration drops to a level that is insufficient to prevent ice/snow build-up (below the solution point in the phase diagram). Once precipitation stops, water or solution remains on the road surface but continues to drain from the road surface or evaporate (process not shown in figure). The \textit{ChemConc} algorithm has been designed to capture the essential elements of the chemical application/dilution process.

\textbf{C.5. Algorithm Description}

The \textit{ChemConc} algorithm is one of several components within the RCTM. The purpose of the \textit{ChemConc} algorithm is to track the forecasted level of anti-icing chemicals (dry or in
solution) and water on the road surface. The \textbf{ChemConc} routine can be triggered in three ways. First, the RCTM may recommend a new treatment, in this case \textit{ChemConc} begins processing at the treatment timestep, calculates the net rate of chemical delivery at the treatment hours and then reduces the available chemical and water over time based on dilution, traffic, runoff and evaporation. Similarly, \textit{ChemConc} may also be triggered by a user selected treatment requiring the same steps as above. Finally, \textit{ChemConc} will be called each time the RWFS weather forecast variables are updated. In this case, \textit{ChemConc} starts at the beginning timestep and applies chemicals based on the current treatment plan. Any time the \textit{ChemConc} algorithm is applied the pavement temperature model is also re-run because the effectiveness of chemicals impacts the level of snow or ice on the road surface, thereby changing the thermal exchange at the road surface.

![Figure C3. Road condition treatment module (RCTM) component algorithms.](image)

There are a number of user parameters that may be used to adjust the sensitivity and thresholds of the algorithm for specific operations. These parameters are described in detail in section \textbf{C.5.c} and are referred to as necessary in the following discussion.

The overall functional flow of the algorithm is shown in Figure C4. The algorithm is currently triggered by the RCTM when a user requests treatment options for a specific route and chemical type (\textit{UserChemType}). The necessary weather variables, road conditions, and treatment recommendations are then sent to \textit{ChemConc}. If a new treatment is to be applied, then the process starts at the treatment application time as opposed to the start of the forecast period. \textit{ChemConc} then loops over each forecast time step calculating

92
the chemical concentration at each successive hour by performing the steps described below.

First, the amount of available water is calculated by summing the water left on the road surface in the previous hour (even if it is already in solution) with the precipitation that fell in the current hour. Parameters are used to control the rate at which new water (Road-WaterImmediateRunoff) and old solution (RoadChemSolutionRunoffPerHour) drain from the road.

Next, there are a series of steps to calculate the amount of available chemicals on the pavement. If this time step is the treatment hour then the ChemRate is set to the recommended treatment application (otherwise it is zero). ChemRate is then adjusted to account for the fraction of the chemical that will be lost due to road splatter and wind effects (RoadChemAppInitialLossRate). If a previous application of chemical was already made then the process determines the amount of residual chemicals - (ResidualChemicals: chemicals that were not dissolved previously because the chemical concentration was above saturation). Finally, the process determines the amount of chemicals already in solution from previous applications (ChemInSolution). The amount of chemicals available for anti-icing operations is defined as:

\[
\text{AvailableChemical} = (1-tfactor)\ast(\text{ChemRate} + \text{ResidualChemical} + \text{ChemInSolution})
\]

where \( tfactor \) represents the fraction of chemicals lost from the road surface due to transport from automobiles and trucks on the road (as calculated in the routine \text{CalcTrafficFactor}). The nominal impact of traffic on available chemicals is controlled by the parameter \text{NominalTrafficDilution}. Increases in traffic levels (Low=1, Medium=2, High=3) results in exponential increases from the nominal dilution value (e.g., a high traffic count is nine times more diluting than a low traffic count).

The next series of steps is used to calculate the final concentration level of the solution remaining on the road surface. The nominal chemical concentration is determined by simply dividing the available chemicals by the sum of the available water and chemicals. However, chemicals can only dissolve into water up to their saturation level. Therefore, the nominal concentration is clipped to the chemical saturation point (as calculated in \text{CalcCriticalChemSaturationPoint}) and used as the final chemical concentration value. Once precipitation has ceased, the available water is allowed to evaporate from the road surface, eventually the road surface will be considered dry and, therefore safe from re-freezing.

One last step is to determine at what time step the chemical concentration will become ineffective. The algorithm currently determines this by finding the first time step where the final chemical concentration is at or below the chemical solution point (as calculated in \text{CalcCriticalChemSolutionPoint}). The forecasted chemical concentration levels and failure time step are then passed back to the RCTM.
Figure C4. Functional flow diagram of ChemConc algorithm.
C.5.a. Algorithm Inputs

The *ChemConc* algorithm requires a forecast of the precipitation rate expressed as a liquid-water equivalent (for example 10" of snow may equal 1.0 " of water). Pavement temperature is another necessary input. If the user is applying a new treatment, then the algorithm needs to know the type, rate and application time of the chemical. The only other input needed is a prediction of the traffic intensity over the forecast period (low, medium or high).

C.5.b. Algorithm Outputs

The *ChemConc* algorithm outputs an estimate of the predicted chemical concentration over the forecast period. In addition, the algorithm predicts the time step where the chemical concentration will become ineffective at preventing ice build-up.

C.5.c. Algorithm Structures

The ChemicalType structure defined in RulesOfPractice.hh holds the parameters/settings applicable to the ChemConc algorithm. The following members are in the structure:

- **UserChemType:** Specify the type of anti-icing (de-icing) chemical to calculate concentrations. There are currently three types supported: 1 – NaCl (Salt), 2- CaCl$_2$ (calcium chloride) and 3 – MgCl$_2$ (magnesium chloride). Other chemicals are not yet fully functional (default is \texttt{P_user_chemtype}).

- **ChemName:** The name of the de-icing chemical (eg. NaCl).

- **Verbose:** Enables (or disables) additional print statements within the code that allow users to see intermediate results and stored values. (True: 1 False: 0).

- **NominalTrafficDilution:** The fractional loss of chemicals (off the roadway) caused by automobile and truck traffic on the road. This nominal value is applied to the lowest level of traffic rates (1=low). Increasing traffic (2=med, 3 =high) results in a squared increase of the traffic impact. So a high traffic roadway experiences 9 times the loss of a low traffic roadway. (nominally set to \texttt{P_NominalTrafficDilution}).

- **RoadChemAppInitialLossRate:** The fractional loss rate of chemicals as they are being applied. This is primarily due to chemicals bouncing off the road surface, but may also be caused by high winds. (nominally set to \texttt{P_InitialChemAppLoss}).

- **RoadWaterImmediateRunoff:** The fractional loss rate of liquid water (runoff) from the road surface in the hour that the precipitation falls (nominally set to \texttt{P_RoadwayImmediateRunoff}).
RoadChemSolutionRunoffPerHour: The fractional loss rate of chemical solution (runoff) at each time step. An assumption is made that the water and chemicals in the solution runoff or dissipate at the same rate. (nominally set to P_ChemSolutionRunoff).

MinApplicationRate: The minimum rate at which the chemical should be applied (nominally set to P_MIN_TREAT).

MaxApplicationRate: The maximum rate at which the chemical should be applied (nominally set to P_MAX_TREAT).

RoundTreatment: When recommending treatment rates, round to this nearest interval (nominally set to P_ROUND_TREAT).

MinTemp: The minimum temperature at which this chemical should be applied (nominally set to the ½ point temperature of the chemical C_NACL_HALFPOINT_TEMP for example).

MaxTemp: The maximum temperature at which this chemical should be applied (nominally set to P_MAX_TEMP).

ChemSafeTempDelta: A sensitivity adjustment (deg C) for the eutectic curve (nominally set to P_CHEM_SAFE DELTA).

C.5.d. Algorithm Parameters

P_user_chemtype: Specify the type of anti-icing (de-icing) chemical to calculate concentrations. There are currently three types supported: 1 – NaCl (Salt), 2- CaCl2 (calcium chloride) and 3 – MgCl2 (magnesium chloride). Other chemicals are not yet fully functional.

P_TREAT_UNITS: The name of the units being used within the algorithm (e.g. “Lbs/lane-mile”)

P_MIN_TREAT: The minimum rate at which the chemical should be applied (in P_TREAT_UNITS).

P_MAX_TREAT: The maximum rate at which the chemical should be applied (in P_TREAT_UNITS).

P_ROUND_TREAT: When recommending treatment rates, round to this nearest interval (in P_TREAT_UNITS).

P_InitialChemAppLossRate: The fractional loss rate of chemicals as they are being applied. This is primarily due to chemicals bouncing off the road surface, but may also be caused by high winds.

P_RoadwayImmediateRunoff: The fractional loss rate of liquid water (runoff) from the road surface in the hour that the precipitation falls

P_RoadwayMinDepthforRunoff: <Not used> Minimal water/solution depth for runoff to occur (mm).
P_ChemSolutionRunoff: The fractional loss rate of chemical solution (runoff) at each time step. An assumption is made that the water and chemicals in the solution runoff or dissipate at the same rate.

P_NominalTrafficDilution: The fractional loss of chemicals (off the roadway) caused by automobile and truck traffic on the road. This nominal value is applied to the lowest level of traffic rates (1=low). Increasing traffic (2=med, 3 =high) results in a squared increase of the traffic impact. So a high traffic roadway experiences 9 times the loss of a low traffic roadway.

P_MIN_TEMP: The minimum temperature at which this chemical should be applied (nominally estimated to the ½ point temperature of the chemical C_NACL_HALFPOINT_TEMP for example).

P_MAX_TEMP: The maximum temperature at which this chemical should be applied (deg C).

P_CHEM_SAFE_DELTA A sensitivity adjustment (deg C) for the eutectic curve.

Table C1 Nominal algorithm parameter values and ranges.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Nominal Value</th>
<th>Units</th>
<th>Expected Range</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_user_chemtype</td>
<td>1</td>
<td>Unitless</td>
<td>1,2,3</td>
<td>1</td>
</tr>
<tr>
<td>P_TREAT_UNITS</td>
<td>“lbs/lane mile”</td>
<td>Character</td>
<td>30 characters</td>
<td>--</td>
</tr>
<tr>
<td>P_MIN_TREAT</td>
<td>100</td>
<td>Lbs/lanemile</td>
<td>0-9999</td>
<td>1</td>
</tr>
<tr>
<td>P_MAX_TREAT</td>
<td>2000</td>
<td>Lbs/lanemile</td>
<td>0-9999</td>
<td>1</td>
</tr>
<tr>
<td>P_ROUND_TREAT</td>
<td>25</td>
<td>Lbs/lanemile</td>
<td>0-500</td>
<td>1</td>
</tr>
<tr>
<td>P_InitialChemAppLossRate</td>
<td>0.10</td>
<td>Fraction</td>
<td>0.0-1.0</td>
<td>0.01</td>
</tr>
<tr>
<td>P_RoadwayImmediateRunoff</td>
<td>0.98</td>
<td>Fraction</td>
<td>0.0-1.0</td>
<td>0.01</td>
</tr>
<tr>
<td>P_ChemSolutionRunoff</td>
<td>0.15</td>
<td>Fraction</td>
<td>0.0-1.0</td>
<td>0.01</td>
</tr>
<tr>
<td>P_RoadwayMinDepthforRunoff</td>
<td>0.05</td>
<td>Fraction</td>
<td>0.0-1.0</td>
<td>0.01</td>
</tr>
<tr>
<td>P_DryChem_LossRate</td>
<td>0.25</td>
<td>Fraction</td>
<td>0.0-1.0</td>
<td>0.01</td>
</tr>
<tr>
<td>P_NominalTrafficDilution</td>
<td>0.01</td>
<td>Fraction</td>
<td>0.0-1.0</td>
<td>0.01</td>
</tr>
<tr>
<td>P_MIN_TEMP</td>
<td>-10.0</td>
<td>Degrees C</td>
<td>-100 to 0</td>
<td>0.1</td>
</tr>
<tr>
<td>P_MAX_TEMP</td>
<td>2.0</td>
<td>Degrees C</td>
<td>-10 to 10</td>
<td>0.1</td>
</tr>
<tr>
<td>P_CHEM_SAFE_DELTA</td>
<td>2.0</td>
<td>Degrees C</td>
<td>-5 to 5</td>
<td>0.01</td>
</tr>
</tbody>
</table>
C.5.e. Algorithm Constants

C_BAD_TEMP, C_BAD_CONC: Bad value flags for concentrations, temperatures and other values. [unitless][-999].

C_mmHr_to_LbSf: Conversion factor for rain from mm/hour to lbs/square-foot. [(mm/hr)/(lb/sf)][0.2047].

C_NOMINAL_LANEWIDTH: The width of a single lane of roadway. [feet][12].

C_MILES_TO_FEET: The number of feet in a mile. [feet][5280].

C_CHEMTYPE_NACL: Internal identifier for salt. [unitless][1].
C_CHEMTYPE_CACL2: Internal identifier for calcium chloride. [unitless][2].
C_CHEMTYPE_MGCL2: Internal identifier for magnesium chloride. [unitless][3].
C_CHEMTYPE_CMA: Internal identifier for calcium magnesium acetate. [unitless][4].
C_CHEMTYPE_KAC: Internal identifier for potassium acetate. [unitless][5].

C_NACL_EUTECTIC_TEMP: The eutectic temperature for salt. [degC][-21.6].
C_CACL2_EUTECTIC_TEMP: The eutectic temperature for calcium chloride. [degC][-51].
C_MGCL2_EUTECTIC_TEMP: The eutectic temperature for magnesium chloride. [degC][-33].
C_CMA_EUTECTIC_TEMP: The eutectic temperature for calcium magnesium acetate. [degC][-27.5].
C_KAC_EUTECTIC_TEMP: The eutectic temperature for potassium acetate. [degC][-60].

C_NACL_EUTECTIC_CONC: The eutectic chemical concentration for salt. [%][23.3].
C_CACL2_EUTECTIC_CONC: The eutectic chemical concentration for calcium chloride. [%][29.8].
C_MGCL2_EUTECTIC_CONC: The eutectic chemical concentration for magnesium chloride. [%][21.6].
C_CMA_EUTECTIC_CONC: The eutectic chemical concentration for calcium magnesium acetate. [%][32.5].
C_KAC_EUTECTIC_CONC: The eutectic chemical concentration for potassium acetate. [%][49.0].

C_NACL_HALFPOINT_TEMP: The temperature on the solution point curve that corresponds to 50% of the eutectic concentration. Used as a measure of the minimum effective temperature of NaCl. [degC][-10.0].
C_CACL2_HALFPOINT_TEMP: The temperature on the solution point curve that corresponds to 50% of the eutectic concentration. Used as a measure of the minimum effective temperature of CaCl₂. [degC][-12.0].

C_MGCL2_HALFPOINT_TEMP: The temperature on the solution point curve that corresponds to 50% of the eutectic concentration. Used as a measure of the minimum effective temperature of MgCl₂. [degC][-9.5].

C_CMA_HALFPOINT_TEMP: The temperature on the solution point curve that corresponds to 50% of the eutectic concentration. Used as a measure of the minimum effective temperature of CMA. [degC][-7.5].

C_KAC_HALFPOINT_TEMP: The temperature on the solution point curve that corresponds to 50% of the eutectic concentration. Used as a measure of the minimum effective temperature of KAC. [degC][-16.0].

C_MIN_ACCEPTABLE_CONC: Used as a minimal chemical concentration value for a chemical to be effective when the temperature is at or above 0 deg C (used in CalcCriticalChemSolutionPoint).

C_CHEM_CHK: Threshold used in RulesOfPractice.cc in situations where plow treatments may be recommended. (P_TREAT_UNITS).

C_NOCHEM_THRESH: Threshold for zeroing out chemical concentration. (lbs/sf)

C_AVAILH20_THRESH: Threshold for zeroing out available roadway water. (lbs/sf).

C_MAX_CHEMNAME: Maximum length of the character string containing the name of the chemical being used (e.g. “Magnesium Chloride”).

C_MAX_TREATNAME: Maximum length of the character string containing the chemical rate units (e.g. “Lbs/lane-mile”)

C_STATUS_OK: Return code that the algorithm is working properly.

C_STATUS_BAD: Return code that the algorithm is not working properly.

References


Appendix D: Net Mobility Module

The DOT users indicated a desire to have a single (non-dimensional) metric to identify the predicted state of the roadway relative to winter road conditions. For demonstration purposes, a simple mobility metric has been developed that takes into account pavement condition (wet, dry, snow, snow depth, ice, etc.).

The Net Mobility Module reads in the meteorological and road surface conditions and outputs an index describing the amount of mobility a vehicle could encounter on the road. This index ranges from 0 (no mobility) to 1 (optimal road conditions). A number of tables exist which describe the mobility in certain conditions. A decision tree leads to finding the proper table. The mobility index is found by finding the proper cell in the table that fits the existing environmental and traffic conditions.

Development Status: This metric needs additional development, as it does not currently take into account some of the subtle factors (e.g., wet snow, dry snow, snow on ice, etc.) that impact mobility. A simple algorithm has been implemented as a placeholder. The algorithm uses the following general scheme:

<table>
<thead>
<tr>
<th>Pavement Condition</th>
<th>Mobility Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dry</td>
<td>1.0</td>
</tr>
<tr>
<td>Wet</td>
<td>0.7</td>
</tr>
<tr>
<td>Snow &lt; 4 inches</td>
<td>0.6</td>
</tr>
<tr>
<td>Snow 4-6 inches</td>
<td>0.4</td>
</tr>
<tr>
<td>Snow &gt; 6 inches</td>
<td>0.3</td>
</tr>
<tr>
<td>Ice</td>
<td>0.2</td>
</tr>
</tbody>
</table>

The utility of using a mobility metric must still be investigated. It is anticipated that several iterations with the users and experience will be required to firm up this concept.

The net mobility code is part of the Road Condition and Treatment Module. The code can be found on the public domain CD in the directory /src/apps/road_cond. The file of interest is calc_mobility.cc.
Appendix E: Rules of Practice Module

The “rules of practice” code is part of the Road Condition and Treatment Module. The code can be found on the public domain CD in the directory /src/apps/road_cond. The files of interest are RulesOfPractice.cc and RulesOfPractice.hh.

E.1. Algorithm Identifier

RulesOfPractice

E.2. Type

Process

E.3. Software

RulesOfPractice.cc

RulesOfPractice.hh

E.4. Conceptual Overview

The RulesOfPractice algorithm is designed to recommend appropriate road treatment actions for winter storm maintenance crews during winter storms. The initial guidance for anti-icing operations were detailed in the FHWA “Manual of Practice for an Effective Anti-icing program: A guide for Highway Maintenance Personnel”. The series of FHWA look-up tables (example shown in Figure E1), however, were based on the eutectic curves of anti-icing chemicals (for further discussion on Eutectic curves see the ChemConc algorithm description Appendix C). Earlier versions of the RulesOfPractice used simplified curves to essentially automate the treatment look-up from the FHWA tables. This latest version for Release 3, however, utilizes both the Eutectic curves directly and the dilution algorithm from the ChemConc algorithm to more accurately estimate the amount of chemicals needed on the road surface to keep snow and ice from bonding.
Maintenance personnel typically gauge the amount of chemical needed during a storm based on past experience and forecasted road and weather conditions. Anti-icing refers to a snow and ice control practice of applying chemicals to prevent the formation or bonding of snow and ice to road surfaces. Treatments are often applied in advance of the actual storm event so that the initial snow/ice does not form a pack on the road surface. The RulesOfPractice algorithm embraces the concept of anti-icing, but also recognizes that some storm conditions (overwhelming snow) or circumstances (equipment breakdown, inadequate crew availability) may necessitate de-icing (the practice of combating the storm as the storm happens -- plowing and applying chemicals as possible to minimize snow and ice build-up). This algorithm is only a first step towards a fully automated guidance system, many simplifications have been made to make this initial task more manageable.

As shown in the functional flow diagram for the RCTM system (Figure C3), the RulesOfPractice module (labeled “Determine Recommended Treatment” box in Figure C3) ingests the ambient weather forecast, the forecasted road conditions and the storm characterization (detailed in Appendix F). The algorithm iterates through the forecast period and identifies the point at which the road surface will become unsafe (called the trigger point). The rules (detailed below) are used to determine the number and type of treatments needed to protect the road surface both during the storm and until the road surface is considered “dry”. Often, treatments will be recommended before the trigger point to protect the road from freezing as the precipitation starts. The algorithm outputs a structure containing the treatments that the system has determined are needed to protect the road surface from re-freezing. Users may select these recommended treatments, or input

Figure E1. Anti-icing treatment table from FHWA guidelines (sample).
their own user-selected treatments. Once a user selects a treatment plan, the RulesOfPractice will only recommend additional treatments if the chosen treatments do not appear (to the automated system) to be sufficient to protect the road surface.

E.5. Algorithm Description

The RulesOfPractice algorithm is one of several key components within the RCTM (Road Condition Treatment Module). The RCTM delivers forecasts of the necessary weather variables (precipitation rate and type – a pass through from the RWFS) road condition variables (pavement temperature and snow depth from the pavement temperature model) and storm characterization. The user selects the type of chemical they want to use to treat the road (UserChemType). The various inputs are passed into the RulesOfPractice algorithm, which in turn looks at the road condition data to determine if treatment may be needed (a build-up of snow or freezing rain on the road for example).

There are a number of user parameters that may be used to adjust the sensitivity and thresholds of the algorithm for specific operations. These parameters are described in detail in section E.5.c and are referred to as necessary in the following discussion.

The overall functional flow of the algorithm is shown in Figure E3.

---

Figure E3  Functional flow diagram for Rules of Practice algorithm.
The algorithm is currently triggered by the RCTM when a user requests treatment options for a specific route and chemical type (UserChemType). The necessary weather variables, road conditions and a storm characterization are then sent to the RulesOfPractice. Currently, three chemicals are supported: 1) NaCl sodium chloride (salt), 2) CaCl₂ – calcium chloride, and 3) MgCl₂ – magnesium chloride.

Next, the algorithm loops over all the forecast hours looking for a treatment trigger. The basic trigger is when a forecasted time period indicates that the frozen precipitation on the road exceeds the bare pavement level of service (r->Bare_Pavement_Thresh). This trigger means that either snow or freezing rain is building up on the road surface. Freezing rain (at any level) is also a trigger point. A routine called DeterminePrecipType is used to determine the current precipitation type (This process is run as part of the CharacterizeStorm module). The RWFS forecast may indicate rain or freezing rain, but the road surface must also be cold enough to support a freezing rain declaration (the parameter P_FRZRAIN_FRZRAIN_TROAD and P_FRZRAIN_RAIN_TROAD control the threshold points for freezing rain). Once a trigger is found, the algorithm attempts to determine the next appropriate treatment. When this first treatment is determined the algorithm applies this single treatment and invokes the ChemConc and Pavement Temperature algorithms to update the road conditions. The algorithm then iterates to find the next treatment (if necessary) until the entire storm has been treated.

The first step in assessing treatment options is to determine if the storm conditions are such that only plowing should be performed. There are several storm criteria that are used to determine if plow-only treatments should be applied, including:

- In-storm road temperatures are Cold or In-range (for chemicals) and becoming too Cold for chemicals
- Road temperatures are In-range or Cold and becoming In-range and significant amounts of blowing snow are forecasted.
- Post-storm road temperatures turn too Cold for chemicals.

Next, if anti-icing chemicals are warranted, the algorithm determines if a pre-treatment strategy should be undertaken. Pre-treatments are performed if the following criteria are met:

- Storm phase types that start as freezing rain.
- Storm starts as snow and:
  - Road temperatures are NOT warm prior to the start of storm, or at the beginning of the storm (road trend type W or WI)
  - Road temperatures are expected to be In-range (I or IW)
  - Road temperatures are cold (C, IC, CI) and significant amounts of blowing snow are NOT forecasted.
- Storm phase type does NOT begin as Rain.

Finally, if pre-treatment isn’t needed (or has already been applied), the RulesOfPractice utilizes the routine CalcChemicalRate() to directly calculate the amount of anti-icing chemicals needed to protect the road surface over the route time (which includes the time to
travel the route and re-supply). In short storms (or near the end of storms), if possible, the chemical rate is increased to cover the remaining storm length.

Once a treatment type and level has been determined, the algorithm must determine the appropriate time to apply the treatment.

For plow-only operations, plowing is recommended at the point where plowable snow is on the road (P_SNOW_PLOW_THRESH). If the threshold is never met, no treatment will be recommended for this route. There is also a user adjustable parameter, P_PLOW_OFFSET, that allows the user to subtract an offset form this hour.

Pre-treatment operations are given a specific time in the MDSS, although this time generally refers to starting the operation on or before the recommended time. For pre-treatments the estimated treatment is changed to a brine recommendation (P_LIQUID_BRINE_EQUIV lbs/lane-mile) and the recommended treatment time is set to the trigger time minus P_LIQUID_BRINE_OFFSET (hours).

For normal anti-icing operations, the application rate is first checked to ensure that it is within limits. If the treatment rate is below the user’s specified minimum (MinApplicationRate), then the MinApplicationRate is recommended instead. Conversely, if the treatment rate is above the user’s specified maximum (MaxApplicationRate) then the recommended rate is capped at MaxApplicationRate. In addition, the recommended start of treatment time is offset backwards from the trigger point by half of the route trip time (RouteRoundtripTiming). This offset allows the application to be applied to most of the route before conditions deteriorate.

E.5.a Algorithm Inputs

The RulesOfPractice algorithm ingests forecasts of precipitation rate and type and pavement temperature that allow the algorithm to adjust the slopes and limits of the recommended treatment curves. Snow depth is used as both the chemical treatment investigation and plowing recommendations. The user must supply the type of chemical they will be using so that the algorithm will only recommend chemical treatments in the appropriate temperature ranges and with reasonable application rates.

E.5.b Algorithm Outputs

The RulesOfPractice algorithm outputs a structure detailing the recommended treatment plan. The structure indicates whether chemical treatment and/or plowing are necessary and at what hour. If chemicals are recommended, the algorithm also returns the chemical type and application rate.

E.5.c Algorithm Structures

There are three structures that are used within the Rules of Practice algorithm: ChemicalType, ROPParamsStruct, TreatmentCurves, and Treatment structures. The
ChemicalType structure is used to define the characteristics of the users preferred anti-icing chemical. This structure is detailed in Appendix C.

The ROPParamsStruct structure contains parameters that help to define the trigger points and route timing for rules of practice decisions. The structure contains:

- **Verbose**: Enables (or disables) additional print statements within the code that allow users to see intermediate results and stored values.
- **Bare_Pavement_Thresh**: The chemical treatment threshold trigger for snow on the road. When a snow depth above the minimum is found, chemical treatment options are investigated.
- **Snow_Plow_Thresh**: The snow plow treatment threshold for snow on the road. Plowing is recommended when snow depths meet or exceed this threshold.
- **RouteRoundTripTiming**: Estimated time it takes to traverse the specified service route. This time delta is used both to determine the length of time to investigate treatments and to estimate the proper pre-treatment time.
- **TruckTurnAroundTime**: Estimated time (in minutes) it takes for a truck to load or reload the anti-icing chemicals.

Finally, the Treatment structure holds all the information about a recommended treatment. The members are:

- **StartTime**: The recommended start time of the plowing or application (in hours from the start of the forecast).
- **EndTime**: The expected end time of the treatment given the route time and truck turn around estimate (in hours from the start of the forecast).
- **EffectiveTime**: The last time period covered by this treatment (in hours) this would be the same as EndTime for a plowing operation. In the case of chemical application, EffectiveTime represents the last time period that the chemicals will be effective as a deicing agent.
- **DoPlowing**: Flag indicating whether the roads should be plowed (1) or not (0).
- **ApplyChemicals**: Flag indicating whether chemicals should be applied (1) or not (0).
- **Chemicals**: The type of chemical to apply (ChemicalType).
- **ApplicationRate**: The rate (in P_TREAT_UNITS) at which to spread the recommended chemical.

**E.5.d. Algorithm Parameters**
P_USER_CHEMTYPE: Specify the type of anti-icing (de-icing) chemical to calculate concentrations. There are currently three types supported: 1 – NaCl (Salt), 2- CaCl₂ (calcium chloride) and 3 – MgCl₂ (magnesium chloride). Other chemicals are not yet fully functional.

P_TREAT_UNITS: The name of the units being used within the algorithm (e.g. “Lbs/lane-mile”)

P_MIN_TREAT: The minimum rate at which the chemical should be applied (in P_TREAT_UNITS).

P_MAX_TREAT: The maximum rate at which the chemical should be applied (in P_TREAT_UNITS).

P_ROUND_TREAT: When recommending treatment rates, round to this nearest interval (in P_TREAT_UNITS).

P_TURNAROUND_TIME: The time it takes to reload a truck (minutes).

P_MIN_TEMP: The minimum temperature at which this chemical should be applied (nominally est to the ½ point temperature of the chemical C_NACL_HALFPOINT_TEMP for example).

P_MAX_TEMP: The maximum temperature at which this chemical should be applied (deg C).

P_CHEM_SAFE_DELTA: A sensitivity adjustment (deg C) for the eutectic curve.

P_BARE_PAVEMENT_THRESH: The chemical treatment threshold trigger for snow on the road. When a snow depth above the minimum is found, chemical treatment options are investigated (inches).

P_SNOW_PLOW_THRESH: The snow plow treatment threshold for snow on the road. Plowing is recommended when snow depths meet or exceed this threshold (inches).

P_FRZRAIN_FRZRAIN_TROAD: The pavement temperature necessary to declare freezing rain on the road surface if the MDSS forecast indicating a precipitation type of freezing rain (deg F).

P_FRZRAIN_RAIN_TROAD: The pavement temperature necessary to declare freezing rain on the road surface if the MDSS forecast is indicating a precipitation type of rain (deg F).

P_LIQUID_BRINE_EQUIV: Current system only allows one type of chemical application (lbs/lane-mile). In order to represent the a typical 50 gal/lane-mile liquid brine treatment we must use a dry salt lbs/lane-mile equivalent of 110 lbs/lane-mile.

P_LIQUID_BRINE_OFFSET: The time offset for pre-treatments is generally different than that of regular applications (it is generally applied much earlier) (hours). This is also used to determine the minimum number of hours of “dry” roads needed to declare a pre-treatment.

Table E1 Nominal algorithm parameter values and ranges.
<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Nominal Value</th>
<th>Units</th>
<th>Expected Range</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_USER_CHEM_TYPE</td>
<td>1</td>
<td>Unitless</td>
<td>1,2,3</td>
<td>1</td>
</tr>
<tr>
<td>P_TREAT_UNITS</td>
<td>&quot;lbs/lanemile&quot;</td>
<td>Character</td>
<td>30 characters</td>
<td>--</td>
</tr>
<tr>
<td>P_MIN_TREAT</td>
<td>100</td>
<td>Lbs/lanemile</td>
<td>0-9999</td>
<td>1</td>
</tr>
<tr>
<td>P_MAX_TREAT</td>
<td>2000</td>
<td>Lbs/lanemile</td>
<td>0-9999</td>
<td>1</td>
</tr>
<tr>
<td>P_ROUND_TREAT</td>
<td>25</td>
<td>Lbs/lanemile</td>
<td>0-500</td>
<td>1</td>
</tr>
<tr>
<td>P_TURNAROUND_TIME</td>
<td>30</td>
<td>Minutes</td>
<td>0-60</td>
<td>1</td>
</tr>
<tr>
<td>P_MIN_TEMP</td>
<td>-10.0</td>
<td>Degrees C</td>
<td>-100 to 0</td>
<td>0.1</td>
</tr>
<tr>
<td>P_MAX_TEMP</td>
<td>2.0</td>
<td>Degrees C</td>
<td>-10 to 10</td>
<td>0.1</td>
</tr>
<tr>
<td>P_CHEM_SAFE_DELTA</td>
<td>2.0</td>
<td>Degrees C</td>
<td>-5 to 5</td>
<td>0.01</td>
</tr>
<tr>
<td>P_BARE_PAVEMENT_THRESH</td>
<td>0.005</td>
<td>Inches</td>
<td>0.001-0.5</td>
<td>0.001</td>
</tr>
<tr>
<td>P_SNOW_PLOW_THRESH</td>
<td>3.0</td>
<td>Inches</td>
<td>0.05-10.0</td>
<td>0.01</td>
</tr>
<tr>
<td>P_FRZRAIN_FRZRAIN_TROAD</td>
<td>35.0</td>
<td>Degrees F</td>
<td>25-40</td>
<td>0.1</td>
</tr>
<tr>
<td>P_FRZRAIN_RAIN_TROAD</td>
<td>35.0</td>
<td>Degrees F</td>
<td>25-40</td>
<td>0.1</td>
</tr>
<tr>
<td>P_LIQUID_BRINE_EQUIV</td>
<td>110.0</td>
<td>lbs/lanemile</td>
<td>110</td>
<td>1</td>
</tr>
<tr>
<td>P_LIQUID_BRINE_OFFSET</td>
<td>6</td>
<td>Hours</td>
<td>0-24</td>
<td>1</td>
</tr>
</tbody>
</table>

**E.5.e. Algorithm Constants**

Following are general constants:
- C_BAD_VAL: Bad value flags for concentrations, temperatures and other values. [unitless][-999].
- C_InHr_to_LbSf: Conversion factor for rain from inches/hour to lbs/square-foot. [((in/hr)/(lb/sf))[5.2].
- C_NOMINAL_LANEWIDTH: The width of a single lane of roadway. [feet][12].
- C_MILES_TO_FEET: The number of feet in a mile. [feet][5280].

Following constants are used to index the type of chemical:
- C_CHEMTYPE_NACL: Internal identifier for salt. [unitless][1].
- C_CHEMTYPE_CACL2: Internal identifier for calcium chloride. [unitless][2].
- C_CHEMTYPE_MGCL2: Internal identifier for magnesium chloride. [unitless][3].

Following constants are used to index the type of treatment recommendation:
- C_TREAT: Apply chemicals (2).
- C_TREAT_PLOWONLY: Plow only (1).
- C_NOTREAT_NONEED: No chemical treatment needed because there is no snow on road (-1).
- C_NOTREAT_TOOCOLD: No chemical treatment needed because it is too cold (-2).
- C_NOTREAT_TOOWARM: No chemical treatment needed because it is too warm (-3).
C_NOTREAT_1515_RULE: No chemical treatment needed because combinations of winds and road temperature will make chemical application hazardous due to blowing snow (-4).

C_NOTREAT_UNDEF: Error in treatment recommendation (-9).

Following constants are used to index the type of precipitation:

- C_PTYPE_NONE: No precipitation (0).
- C_PTYPE_RAIN: Rain (1).
- C_PTYPE_SNOW: Snow (2).
- C_PTYPE_MIXED: Snow/rain mix (3).
- C_PTYPE_SLEET: Sleet (4).
- C_PTYPE_FRZRAIN: Freezing rain (5).

Following are currently constants, but will eventually become parameters as they are used to modify the recommended spread rate based on precip intensity. Moderate yields no change from nominal settings, light reduces recommendation by P_LGT_PRECIP_FACTOR and heavy increases the recommendation by P_HVY_PRECIP_FACTOR.

- P_LGT_PRECIP_RATE: The liquid equivalent rate that corresponds to light precipitation. (inches/hour) (0.10).
- P_MOD_PRECIP_RATE: The liquid equivalent rate that corresponds to moderate precipitation. (inches/hour) (0.25).
- P_HVY_PRECIP_RATE: The liquid equivalent rate that corresponds to heavy precipitation. (inches/hour) (0.75).
- P_RDELTA: Pre-calculated heavy to light rate difference for ratio calculations.

- P_LGT_FRZRAIN_RATE: The liquid equivalent rate that corresponds to light precipitation. (inches/hour) (0.01).
- P_MOD_FRZRAIN_RATE: The liquid equivalent rate that corresponds to moderate precipitation. (inches/hour) (0.03).
- P_HVY_FRZRAIN_RATE: The liquid equivalent rate that corresponds to heavy precipitation. (inches/hour) (0.75).
- P_FRZ_DELTA: Pre-calculated heavy to light freezing rain rate difference for ratio calculations.

- P_LGT_PRECIP_FACTOR: As noted above, light precipitation will reduce the overall treatment recommendations by this fraction. (0.65).
- P_HVY_PRECIP_FACTOR: As noted above, heavy precipitation will increase the overall treatment recommendations by this fraction. (1.35).
- P_FDELTA: Pre-calculated heavy to light factor difference for ratio calculations.
References


Appendix F: Characterize Storm Module

The storm characterization algorithm is part of the Road Condition and Treatment Module. The code can be found on the public domain CD in the directory /src/apps/road_cond. The files of interest are CharacterizeStorm.cc and RulesOfPractice.hh.

F.1. Algorithm Identifier

CharacterizeStorm

F.2. Type

Process

F.3. Software

CharacterizeStorm.cc

RulesOfPractice.hh

F.4. Conceptual Overview

Understanding the overall structure, extent and type of winter storm is an important factor in determining the type and extent (if any) of the treatment required. For example, knowing that a storm will start as rain indicates that pre-treatment should be suppressed, knowing that post-storm temperatures will be below the range chemical effectiveness may indicate that only plowing should be performed. In support of this goal, the CharacterizeStorm algorithm is designed to summarize the characteristics of the overall storm (prior, during and after precipitation) that are relevant to the type and extent of treatment that may be required. Figure C3 shows the overall structure of the RCTM. The CharacterizeStorm algorithm is invoked after the initialization of the road conditions based on the current weather forecast, previous road conditions and operational characteristics of the route. The routine outputs a structure containing information on the pre-, in-, and post-storm environment. Figure F2 illustrates the breakdown of storm characteristics gathered for the pre-, in-, and post-storm environment.
The road temperature trends for assessing chemical effectiveness are represented by “W” for too warm, “I” for in-range, “C” for too cold. Precipitation types are represented by “R” for rain, “Z” for ice, “S” for snow. Combinations of types are also shown.

Starting from the right, the pre-storm environment is captured by measuring the road temperature trends as warm, in-range, or cold. A simple fuzzy logic template is applied to the time period 6 hours prior to the start of measurable precipitation (above 0.01 inches liquid water equivalent) on the road surface. The template applied is shown in figure F3 (a) warm (b) in-range (c) cold. In-range measurements represent road temperatures that are within the effective range of the anti-icing chemical to be used (refer to the ChemConc algorithm description in Appendix C). The road temperature characterization of warm or cold environments indicates time segments where the road temperatures are above or below the effective chemical range, respectively. The fuzzy scoring function is used to reduce the impact of borderline or spurious measurements on the system. Similarly, the road temperature trend is captured for the post-storm environment shown on the left-hand side of Figure F2.
There are several in-storm environment characteristics that are captured by the algorithm. Overall precipitation characteristics, such as total amount and timing (stop, start, and duration) of both liquid and frozen precipitation are captured first. Road temperature trends are also captured, but up to two phases of temperatures are allowed. For example, roads may start out as warm and transition to in-range, or start in-range and turn too cold. There are 7 combinations of road temperatures: W, I, C, WI, IW, IC, CI. Transitions that skip a range are unlikely and are not captured by the system (e.g. WC). Finally, the phase type of the storm is captured. Like the road temperature trends a scoring function is applied to determine the type of storm. Since many storms have multiple phases, up to 3 phase changes are allowed per storm. Therefore the system should be able to identify a storm that starts as rain, turns briefly to freezing rain and then to all snow (RZS). In the case of phase change, skipping a phase is allowed so that a storm may be classified as rain changing to snow (RS). There are 23 classifications of in-storm precipitation phase, as shown in Table F1.

Table F1 Storm phase type identifier and description.

<table>
<thead>
<tr>
<th>Storm phase type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>All rain.</td>
</tr>
<tr>
<td>Z</td>
<td>All freezing rain.</td>
</tr>
<tr>
<td>S</td>
<td>All snow.</td>
</tr>
<tr>
<td>RS</td>
<td>Rain changing to snow.</td>
</tr>
<tr>
<td>RZ</td>
<td>Rain changing to freezing rain.</td>
</tr>
<tr>
<td>SZ</td>
<td>Snow changing to freezing rain.</td>
</tr>
<tr>
<td>SR</td>
<td>Snow changing to rain.</td>
</tr>
<tr>
<td>ZR</td>
<td>Freezing rain changing to rain.</td>
</tr>
<tr>
<td>ZS</td>
<td>Freezing rain changing to snow.</td>
</tr>
<tr>
<td>RRZ</td>
<td>Rain ending as freezing rain.</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>RZZ</td>
<td>Starting as rain turning to freezing rain.</td>
</tr>
<tr>
<td>RZR</td>
<td>Rain with a brief period of freezing rain.</td>
</tr>
<tr>
<td>RZS</td>
<td>Starting as rain, turning to freezing rain and ending as snow.</td>
</tr>
<tr>
<td>ZZR</td>
<td>Freezing rain ending as rain.</td>
</tr>
<tr>
<td>ZZR</td>
<td>Freezing rain ending as snow.</td>
</tr>
<tr>
<td>ZRR</td>
<td>Starting as freezing rain and turning to rain.</td>
</tr>
<tr>
<td>ZRZ</td>
<td>Freezing rain with a brief period of rain.</td>
</tr>
<tr>
<td>ZSS</td>
<td>Starting as freezing rain and turning to snow.</td>
</tr>
<tr>
<td>ZSZ</td>
<td>Freezing rain with a brief period of snow.</td>
</tr>
<tr>
<td>SSZ</td>
<td>Snow ending as freezing rain.</td>
</tr>
<tr>
<td>SZZ</td>
<td>Starting as snow and turning to freezing rain.</td>
</tr>
<tr>
<td>SZS</td>
<td>Snow with a brief period of freezing rain.</td>
</tr>
<tr>
<td>SZR</td>
<td>Starting as snow, turning to freezing rain and ending as rain.</td>
</tr>
</tbody>
</table>

One final derived variable is a measurement of the likelihood that road temperatures will be cold and blowing snow will be present. This measure of blowing is not the same as that captured in the new MDSS blowing snow algorithm, but it will be in the future. The measure is designed to flag storms where treatments should be suppressed in lieu of plow only operations because the snow will simply blow across the road without sticking to the surface. A redesign of this parameter is likely so a detailed description is not given here.

F.5. Algorithm Description

The *StormCharacterization* algorithm, as described above, is a key component of the RCTM is designed to capture the overall storm characteristics (prior, during and after precipitation) that are relevant to the type and extent of treatment that may be required.

There are a number of user parameters that may be used to adjust the sensitivity and thresholds of the algorithm for specific operations. These parameters are described in detail in section F.5.c and are referred to as necessary in the following discussion.

The overall functional flow of the algorithm is shown in Figure C3. The algorithm is currently triggered by the RCTM whenever a new set of forecasted weather conditions is received. The algorithm ingests current weather forecast, previous road conditions and operational characteristics of the route and outputs a structure containing information on the pre-, in-, and post-storm environment.
Figure F4. Functional flow diagram of the CharacterizeStorm algorithm.

F.5.a Algorithm Inputs

The CharacterizeStorm algorithm ingests forecasts of precipitation rate and type, wind speed and pavement temperature. Forecasted precipitation type and pavement temperature are used to determine the expected phase of the any moisture on the road and to capture the hourly extent of the storm event. Pavement temperature alone is used to determine the road temperature trends within the storm, likewise precipitation type and storm type (rain, snow, rain-snow, etc). Forecasts of wind speed are also ingested to determine a crude index of blowing snow potential (this will be replaced in later versions with output from the Blowing Snow algorithm).

F.5.b Algorithm Outputs

The CharacterizeStorm algorithm outputs a structure containing variables that describe the expected type of storm. Pre-, In-, and Post-storm codes for road temperature trends and In-storm characterization of the storm precipitation phase changes. In addition, summary statistics are also saved on storm start and stop and total liquid and frozen precipitation. The structure is detailed in section F5.c.
F.5.c. Algorithm Structures

There is one primary structure used within the CharacterizeStorm algorithm: StormType. The StormType structure holds all the storm summary information gathered in CharacterizeStorm, as follows:

- **StartAnyPrecip**: The first hour that any type of precipitation is forecasted (in hours from the start of the forecast).
- **EndAnyPrecip**: The last hour that any type of precipitation is forecasted (in hours from the start of the forecast).
- **TotalLiquidPrecip**: The total liquid water equivalent (mm).
- **TotalFrozenPrecip**: The total depth of frozen precipitation (mm).
- **StartFrozenPrecip**: The first hour that frozen precipitation is forecasted (in hours from the start of the forecast).
- **EndFrozenPrecip**: The last hour that frozen precipitation is forecasted (in hours from the start of the forecast).
- **RoadTrend_PrestormType**: The road temperature trend category prior to the start of storm precipitation (Warm, In-range, or Cold).
- **RoadTrend_PoststormType**: The road temperature trend category after the end of storm precipitation (Warm, In-range, or Cold).
- **RoadTrend_StormType**: The road temperature trend category during the storm. (Warm, In-range, Cold with up to 2 tiers, e.g. Cold to Warm).
- **PrecipType_StormType**: The phase of precipitation during the storm (Snow, Freezing Rain, or Rain, with up to 3 tiers, e.g. Snow to Rain to Snow).
- **PrecipRate_StormType**: Not Used. Characterize the change in storm intensity.
- **ColdBS_score**: Score to indicate the likelihood that blowing snow will likely fall on cold roads.

F.5.d. Algorithm Parameters

- **P_ROADT_WARM**: Specify the type of anti-icing (de-icing) chemical to calculate concentrations. There are currently three types supported: 1 – NaCl (Salt), 2- CaCl2 (calcium chloride) and 3 – MgCl2 (magnesium chloride). Other chemicals are not yet fully functional.
- **P_ROADT_INRANGE**: The name of the units being used within the algorithm (e.g. “Lbs/lane-mile”)
- **P_ROADT_TOOCOLD**: The minimum rate at which the chemical should be applied (in P_TREAT_UNITS).
- **P_ROADT_NUM_PRE**: The number of hours prior to the start of precipitation that constitutes the pre-storm environment (hours).
- **P_ROADT_NUM_POST**: The number of hours after the end of precipitation that constitutes the post-storm environment (hours).
Table F2 Nominal algorithm parameter values and ranges.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Nominal Value</th>
<th>Units</th>
<th>Expected Range</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{ROADT_WARM}$</td>
<td>10.0</td>
<td>Degrees C</td>
<td>0 to 40</td>
<td>0.1</td>
</tr>
<tr>
<td>$P_{ROADT_INRANGE}$</td>
<td>-5.0</td>
<td>Degrees C</td>
<td>-10 to 0</td>
<td>0.1</td>
</tr>
<tr>
<td>$P_{ROADT_TOOCOLD}$</td>
<td>-15.0</td>
<td>Degrees C</td>
<td>-25 to 0</td>
<td>0.1</td>
</tr>
<tr>
<td>$P_{ROADT_NUM_PRE}$</td>
<td>6</td>
<td>Hours</td>
<td>1-12</td>
<td>1</td>
</tr>
<tr>
<td>$P_{ROADT_NUM_POST}$</td>
<td>6</td>
<td>Hours</td>
<td>1-12</td>
<td>1</td>
</tr>
</tbody>
</table>

F.5.e. Algorithm Constants

Following constants are used to index the type of precipitation:
- C_PTYPE_NONE: No precipitation (0).
- C_PTYPE_RAIN: Rain (1).
- C_PTYPE_SNOW: Snow (2).
- C_PTYPE_MIXED: Snow/rain mix (3).
- C_PTYPE_SLEET: Sleet (4).
- C_PTYPE_FRZRAIN: Freezing rain (5).

References


Appendix G: Precipitation Type Algorithms

The MDSS system includes three algorithms that diagnose the type of precipitation (e.g. rain, snow, freezing rain, and sleet) that is expected given the forecast state of the atmosphere provided by a numerical model. Each of these algorithms determines the most likely type of precipitation at a particular location based on an assessment of a vertical profile of dry-bulb and dewpoint temperatures at that location. Since these algorithms use different formulations for assessing the precipitation type, they have been integrated into the Road Weather Forecast System (RWFS) to provide the MDSS with an approximation of the precipitation-type forecast uncertainty.

The algorithms are written as subroutines that require one-dimensional thermodynamic data at a particular location as input. The data that are needed at each level are pressure, dry-bulb temperature, and dewpoint temperature. There is no minimum requirement for the number of data levels, although fewer than ten would probably reduce the accuracy significantly.

Brief descriptions of the specific algorithms follow. For more details, consult the references in the Precipitation Type References file included in this document.

a) NCEP (also see Baldwin et al. (1994))

The National Centers for Environmental Prediction (NCEP) algorithm currently is being used by the National Weather Service to generate precipitation type data for their forecasters. Using forecast sounding data, this algorithm quantifies the thermodynamic stratification and compares it to a set of empirically-determined set of similar variables to diagnosis the precipitation type.

b) Bourgoin (also see Bourgoin (2000))

The algorithm written by Pierre Bourgoin is currently being used by the Meteorological Service of Canada to generate precipitation-type data for their forecasters. The procedure for computing precipitation type is similar to that used in the NCEP algorithm, except that different empirically-determined values are used.

c) Ramer (also see Ramer (1993))

The Ramer algorithm diagnoses the state of a single hydrometeor as it falls from a generating level to the ground. Using the forecast sounding data, the algorithm computes how much melting and refreezing will occur as the hydrometeor descends through each atmospheric layer using fundamental thermodynamic principles.
**Precipitation Type Algorithm References**


Appendix H: Experimental Blowing Snow Potential Algorithm

An experimental blowing snow potential algorithm was added to the MDSS in 2004 to provide end users with an indication of the likelihood that blowing snow conditions may exist and winter maintenance treatments may have to be performed. A blowing snow indicator function is calculated for each hour of the MDSS 48-hr forecast. The indicator is a continuous function taking on values between zero and one. Thresholds are applied to the indicator function to categorize the blowing snow potential into one of four threat levels at each forecast lead time. The output categories are:

<table>
<thead>
<tr>
<th>Alert Color</th>
<th>MDSS Alert Category</th>
<th>Meaning</th>
<th>Indicator Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>OK</td>
<td>Low Potential</td>
<td>0.00 – 0.05</td>
</tr>
<tr>
<td>Yellow</td>
<td>Marginal</td>
<td>Medium Potential</td>
<td>0.05 – 0.333</td>
</tr>
<tr>
<td>Red</td>
<td>Poor</td>
<td>High Potential</td>
<td>0.333 – 0.667</td>
</tr>
<tr>
<td>Purple</td>
<td>Extreme</td>
<td>Very High Potential</td>
<td>0.667 – 1.00</td>
</tr>
</tbody>
</table>

The blowing snow algorithm is based on four key factors:
1) How recently did it snow or was it forecast to snow?
2) How windy is it at the forecast time?
3) Did liquid precipitation (rain, freezing rain, drizzle, or freezing drizzle) occur in the hours following the snowfall?
4) How warm did it get since it snowed?

Each factor is assigned a value between zero and one. The higher the value, the more likely it is that blowing snow could occur. Basically, the four factors can be described and are calculated as follows:

1) The more recent the snow, the better the chance for blowing snow. When the snow is fresh, it has a better chance of blowing around. Once the snow gets to be more than a day old, the chances for blowing snow decrease. After three days, the chance goes to zero in the algorithm. The factor $F_1$ is calculated as follows in terms of $h$, the number of hours since the last snowfall:

$$F_1 = \begin{cases} 1 & \text{if } h < 12 \\ \frac{(72-h)}{60} & \text{if } 12 \leq h \leq 72 \\ 0 & \text{if } h \geq 72 \end{cases}$$

2) The higher the ‘sustained’ wind speed, the better the chance for blowing snow. 
Wind gusts are estimated to be 50% higher than sustained winds. The general idea is:

Wind speed < 3.75 m/s, chance for blowing snow is ZERO.
3.75 m/s < wind speed < 8.75 m/s, chances increase from "zero" to "high".
Wind speed > 8.75 m/s, chance of blowing snow is "very high".

Mathematically, the factor $F_2$ is calculated as follows in terms of $wspd$, the wind speed in meters per second, at the forecast time:

$$F_2 = \begin{cases} 
0 & \text{if } wspd < 3.75 \\
(wspd-3.75)/5 & \text{if } 3.75 \leq wspd \leq 8.75 \\
1 & \text{if } wspd > 8.75
\end{cases}$$

3) If there was liquid precipitation (rain, freezing rain, drizzle, or freezing drizzle) since the last snowfall, the top of the snow should have melted down some and/or formed a frozen top crust. In this case, the chance for blowing snow is set to ZERO. Factor $F_3$ is calculated as flows:

$$F_3 = \begin{cases} 
0 & \text{if rain or freezing rain has occurred since last snowfall} \\
1 & \text{if snow was the last precipitation type}
\end{cases}$$

4) If it gets warm enough, then the top of the snow will tend to melt and may refreeze. If the air temperature exceeded +1C, then the chances for blowing snow are set to ZERO. If it has been colder than -4C the whole time, then the snow should have been unaffected. Mathematically, factor $F_4$ is calculated as flows in terms of the maximum temperature, $T_x$, since the last snowfall:

$$F_4 = \begin{cases} 
0 & \text{if } T_x > +1C \\
((1 - T_x)/5)^{0.75} & \text{if } -4C \leq T_x \leq +1C \\
1 & \text{if } T_x < -4C
\end{cases}$$

The final blowing snow potential indicator function is a multiplicative combination of the above factors. That is:

$$I = F_1 \times F_2 \times F_3 \times F_4.$$

One effect of this combination method is that the final indicator function’s value will be less than or equal to each of the contributing factors. Therefore if any of the four factors indicates that blowing snow is unlikely to occur, the multiplicative combination will indicate a low likelihood. For high blowing snow potential values to occur, all four factors must be high.

Once this final indicator function is calculated, the range thresholds are applied and the event is categorized as either "OK", "Marginal", "Poor", or "Extreme" blowing snow potential.
Appendix I: Installing RWFS from CD-ROM

Before installing the RWFS software from the CD, note the following:

- This step installs the binaries, scripts and configuration files for the RWFS component of the MDSS system. Additional configuration may be required after this step. See section 12 for further information.
- See the system hardware, software and external (3rd party) software requirements

1. Create, then log into an mdss account
2. Open a text window on your display
3. Load the CD into the CD-ROM drive
4. From the text window, type:

   `% /cdrom/cdrom0/install`

   (NOTE: Your CD-ROM drive path may vary.)

   Follow the instructions on the screen to install the RWFS system.
## Appendix J: Technical Points of Contact

The primary technical points of contact for items contained in the MDSS Release-3 package are listed below.

<table>
<thead>
<tr>
<th>MDSS Technical Component</th>
<th>Source Lab</th>
<th>Technical Point of Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Road Temperature Model SNTHERM-RT</td>
<td>CRREL</td>
<td>Gary Phetteplace&lt;br&gt;CRREL&lt;br&gt;72 Lyme Road&lt;br&gt;Hanover, NH 03755-1290&lt;br&gt;Ph: 603-646-4248&lt;br&gt;Email: <a href="mailto:Gary.E.Phetteplace@erdc.usace.army.mil">Gary.E.Phetteplace@erdc.usace.army.mil</a>&lt;br&gt;George Koenig&lt;br&gt;CRREL&lt;br&gt;72 Lyme Road&lt;br&gt;Hanover, NH 03755-1290&lt;br&gt;Ph: 603-646-4556&lt;br&gt;Fax: 603-646-4730&lt;br&gt;Email: <a href="mailto:gkoenig@crrel.usace.army.mil">gkoenig@crrel.usace.army.mil</a></td>
</tr>
<tr>
<td>Chemical Concentration Algorithms</td>
<td>LL (Algorithm Software)</td>
<td>Robert G. Hallowell&lt;br&gt;MIT Lincoln Laboratory&lt;br&gt;244 Wood Street&lt;br&gt;Lexington MA 02420-9180&lt;br&gt;Ph: 781-981-3645&lt;br&gt;Fax: 781-981-0632&lt;br&gt;Email: <a href="mailto:bobh@ll.mit.edu">bobh@ll.mit.edu</a></td>
</tr>
<tr>
<td>Coded Rules of Practice</td>
<td>LL (Algorithm Software)</td>
<td>Robert G. Hallowell&lt;br&gt;MIT Lincoln Laboratory&lt;br&gt;244 Wood Street&lt;br&gt;Lexington MA 02420-9180&lt;br&gt;Ph: 781-981-3645&lt;br&gt;Fax: 781-981-0632&lt;br&gt;Email: <a href="mailto:bobh@ll.mit.edu">bobh@ll.mit.edu</a></td>
</tr>
<tr>
<td>Road Weather Forecast System (based on DICAST©)</td>
<td>NCAR</td>
<td>Bill Myers, Lead Software Engineer&lt;br&gt;NCAR&lt;br&gt;3450 Mitchell Lane&lt;br&gt;Boulder CO 80301&lt;br&gt;Ph: 303-497-8412&lt;br&gt;Fax: 303-497-8401&lt;br&gt;Email: <a href="mailto:myers@ucar.edu">myers@ucar.edu</a>&lt;br&gt;Jim Cowie&lt;br&gt;NCAR&lt;br&gt;3450 Mitchell Lane&lt;br&gt;Boulder, CO 80301&lt;br&gt;Ph: 303-497-2831&lt;br&gt;Fax: 303-497-8401&lt;br&gt;Email: <a href="mailto:cowie@ucar.edu">cowie@ucar.edu</a></td>
</tr>
<tr>
<td>Road Condition and Treatment Module System Integration</td>
<td>NCAR</td>
<td>Paddy McCarthy&lt;br&gt;NCAR&lt;br&gt;3450 Mitchell Lane&lt;br&gt;Boulder CO 80301&lt;br&gt;Ph: 303-497-8461&lt;br&gt;Fax: 303-497-8401&lt;br&gt;Email: <a href="mailto:paddy@ucar.edu">paddy@ucar.edu</a></td>
</tr>
</tbody>
</table>

124
| Arnaud Dumont | Arnaud Dumont  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NCAR</td>
<td>NCAR</td>
</tr>
<tr>
<td>3450 Mitchell Lane</td>
<td>3450 Mitchell Lane</td>
</tr>
<tr>
<td>Boulder CO 80301</td>
<td>Boulder CO 80301</td>
</tr>
<tr>
<td>Ph: 303-497-8434</td>
<td>Ph: 303-497-8434</td>
</tr>
<tr>
<td>Fax: 303-497-8401</td>
<td>Fax: 303-497-8401</td>
</tr>
<tr>
<td><a href="mailto:dumont@ucar.edu">dumont@ucar.edu</a></td>
<td><a href="mailto:dumont@ucar.edu">dumont@ucar.edu</a></td>
</tr>
<tr>
<td>Patty Miller</td>
<td>Patty Miller</td>
</tr>
<tr>
<td>NOAA/Forecast Systems Lab</td>
<td>NOAA/Forecast Systems Lab</td>
</tr>
<tr>
<td>325 Broadway, R/FS1</td>
<td>325 Broadway, R/FS1</td>
</tr>
<tr>
<td>Boulder CO 80303</td>
<td>Boulder CO 80303</td>
</tr>
<tr>
<td>Ph: 303-497-6365</td>
<td>Ph: 303-497-6365</td>
</tr>
<tr>
<td>Fax: 303-497-7256</td>
<td>Fax: 303-497-7256</td>
</tr>
<tr>
<td>Email: <a href="mailto:Patricia.A.Miller@noaa.gov">Patricia.A.Miller@noaa.gov</a></td>
<td>Email: <a href="mailto:Patricia.A.Miller@noaa.gov">Patricia.A.Miller@noaa.gov</a></td>
</tr>
<tr>
<td>FSL</td>
<td>FSL</td>
</tr>
<tr>
<td>Paul Schultz</td>
<td>Paul Schultz</td>
</tr>
<tr>
<td>NOAA/Forecast Systems Lab</td>
<td>NOAA/Forecast Systems Lab</td>
</tr>
<tr>
<td>325 Broadway, R/FS1</td>
<td>325 Broadway, R/FS1</td>
</tr>
<tr>
<td>Boulder CO 80303</td>
<td>Boulder CO 80303</td>
</tr>
<tr>
<td>Ph: 303-497-6997</td>
<td>Ph: 303-497-6997</td>
</tr>
<tr>
<td>Fax: 303-497-7262</td>
<td>Fax: 303-497-7262</td>
</tr>
<tr>
<td>Email: <a href="mailto:paul.j.schultz@noaa.gov">paul.j.schultz@noaa.gov</a></td>
<td>Email: <a href="mailto:paul.j.schultz@noaa.gov">paul.j.schultz@noaa.gov</a></td>
</tr>
<tr>
<td>FSL</td>
<td>FSL</td>
</tr>
<tr>
<td>Bill Mahoney</td>
<td>Bill Mahoney</td>
</tr>
<tr>
<td>NCAR</td>
<td>NCAR</td>
</tr>
<tr>
<td>3450 Mitchell Lane</td>
<td>3450 Mitchell Lane</td>
</tr>
<tr>
<td>Boulder CO 80301</td>
<td>Boulder CO 80301</td>
</tr>
<tr>
<td>Ph: 303-497-8426</td>
<td>Ph: 303-497-8426</td>
</tr>
<tr>
<td>Fax: 303-497-8401</td>
<td>Fax: 303-497-8401</td>
</tr>
<tr>
<td>Cell: 303-817-7975</td>
<td>Cell: 303-817-7975</td>
</tr>
<tr>
<td>Email: <a href="mailto:mahoney@ucar.edu">mahoney@ucar.edu</a></td>
<td>Email: <a href="mailto:mahoney@ucar.edu">mahoney@ucar.edu</a></td>
</tr>
<tr>
<td>FSL</td>
<td>FSL</td>
</tr>
<tr>
<td>Paul Pisano</td>
<td>Paul Pisano</td>
</tr>
<tr>
<td>FHWA</td>
<td>FHWA</td>
</tr>
<tr>
<td>HOTO-1 Room 3408</td>
<td>HOTO-1 Room 3408</td>
</tr>
<tr>
<td>400 Seventh St SW</td>
<td>400 Seventh St SW</td>
</tr>
<tr>
<td>Washington, D.C. 20590</td>
<td>Washington, D.C. 20590</td>
</tr>
<tr>
<td>Email: <a href="mailto:paul.pisano@fhwa.dot.gov">paul.pisano@fhwa.dot.gov</a></td>
<td>Email: <a href="mailto:paul.pisano@fhwa.dot.gov">paul.pisano@fhwa.dot.gov</a></td>
</tr>
<tr>
<td>FSL</td>
<td>FSL</td>
</tr>
<tr>
<td>Rudy Persaud</td>
<td>Rudy Persaud</td>
</tr>
<tr>
<td>FHWA/HRDO</td>
<td>FHWA/HRDO</td>
</tr>
<tr>
<td>Turner Fairbanks Research Center</td>
<td>Turner Fairbanks Research Center</td>
</tr>
<tr>
<td>6300 Georgetown Pike</td>
<td>6300 Georgetown Pike</td>
</tr>
<tr>
<td>McLean, VA 22101</td>
<td>McLean, VA 22101</td>
</tr>
<tr>
<td>Ph: 202-493-3391</td>
<td>Ph: 202-493-3391</td>
</tr>
<tr>
<td>Email: <a href="mailto:rudy.persaud@fhwa.dot.gov">rudy.persaud@fhwa.dot.gov</a></td>
<td>Email: <a href="mailto:rudy.persaud@fhwa.dot.gov">rudy.persaud@fhwa.dot.gov</a></td>
</tr>
<tr>
<td>FSL</td>
<td>FSL</td>
</tr>
<tr>
<td>Andy Stern</td>
<td>Andy Stern</td>
</tr>
<tr>
<td>Mitretek</td>
<td>Mitretek</td>
</tr>
<tr>
<td>3150 Fairview Park Drive South</td>
<td>3150 Fairview Park Drive South</td>
</tr>
<tr>
<td>MS-530</td>
<td>MS-530</td>
</tr>
<tr>
<td>Falls Church, VA 22042</td>
<td>Falls Church, VA 22042</td>
</tr>
<tr>
<td>Ph: 703-610-1754</td>
<td>Ph: 703-610-1754</td>
</tr>
<tr>
<td>Email: <a href="mailto:astern@mitretek.org">astern@mitretek.org</a></td>
<td>Email: <a href="mailto:astern@mitretek.org">astern@mitretek.org</a></td>
</tr>
</tbody>
</table>