Integrated Modeling for Road Condition Prediction

Installation/Administration Guide

www.its.dot.gov/index.htm
December 2017

U.S. Department of Transportation

Produced under Contract DTFH61-12-D-00045, Support Services for the Office of Operations (HOP) Transportation Operations (HOTO) and Transportation Management (HOTM) Operations Group 2 Operations and Intelligent Transportation Systems (O&ITS)
U.S. Department of Transportation
Office of the Assistant Secretary for Research and Technology, Intelligent Transportation Systems Joint Program Office
Federal Highway Administration, Office of Operations

Notice

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

Background

Transportation systems management and operations (TSMO) is at a critical point in its development due to an explosion in data availability and analytics. New approaches in road weather management are bringing together meteorology, traffic management, law enforcement, maintenance, and traveler information to support agency decision making and influence travel behavior. Through these operational efforts and private sector innovations, travelers today have higher expectations for their travel experience. Travelers now participate in generating and validating information as well as consuming it. This trend will accelerate with deployment of connected vehicle (CV) systems. Within this context, the role of prediction and forecasting will become more important to the travel and activity choices made by travelers, as well as to agency decisions in transportation operations. Freight carriers and logistics providers will also benefit in planning routes, times, and delivery schedules.

Development and adoption of traffic prediction approaches by operating agencies have been limited, however, even with a growing body of research. While this is partly attributable to limited data, available predictive tools have been narrowly focused and have not taken full advantage of developments in related disciplines and domains. As a result, the use of predictive methods in support of operational decisions continues to be limited.

Recent efforts to incorporate forecast weather conditions in traffic predictions have shown considerable promise. Factoring in reported conditions from environmental sensor stations, vehicle fleets, and citizen-reported conditions could improve estimation of the current system state from which predictions are developed. The utility of traffic predictions could be further enhanced by augmenting the forecast weather conditions with known and likely capacity constraints such as work zones and incidents. Current and planned road treatment approaches, snowplow routing, parking restrictions, and maintenance decisions could be included as well.

Based on these opportunities, the Federal Highway Administration (FHWA) has undertaken the investigation, development, and demonstration deployment of an Integrated Model for Road Condition Prediction (IMRCP). This effort has included a survey of available and imminent weather, hydrological, traffic, and related transportation management models; development of a concept of operations (ConOps) and fundamental system requirements; development of a system architecture and system design; implementation of a foundational system; and deployment of the system with an operating transportation agency to evaluate its effectiveness. Research, development, and operations stakeholders have been involved in every part of the IMRCP effort.

Purpose

The purpose of the IMRCP is to integrate weather, traffic, and other operations data sources with analytical methods to effectively predict road and travel conditions. Identification of system functions and interfaces is driven by stakeholders in operations, who also provide feedback on the usefulness of the model results. The model could ultimately become a practical tool for transportation agencies to support traveler advisories, maintenance plans, and operational decisions at both strategic and tactical levels.

The purpose of this Administrator Guide is to document relevant information regarding the installation, configuration, operations, and maintenance of the IMRCP system.

Scope

The IMRCP provides a framework for the integration of road condition monitoring and forecast data to support tactical and strategic decisions by travelers, transportation operators, and maintenance providers. The system collects and integrates environmental observations and transportation operations data; collects forecast environmental and operations data when available; initiates road weather and traffic forecasts based on the collected data; generates travel and operational advisories and warnings from the collected real-time and forecast data; and provides the road condition data, forecasts, advisories, and warnings to other applications and systems. Road condition and operations data and forecasts integrated into the prediction may include: atmospheric weather; road (surface) weather; small stream, river, and coastal water levels; road network capacity; road network demand; traffic conditions and forecasts; traffic control states; work zones; maintenance activities and plans; and emergency preparedness and operations.

Document Overview

The remaining sections of the document contain information on the administration and installation of the IMRCP system.

Chapter 2, Installation describes the server environment as well as installation of IMRCP components.

Chapter 3, Application Configuration describes the configuration file used for the IMRCP system.

Chapter 4, Model Configuration describes the configuration of the road network, traffic demand, road weather, hydrological, Bayes traffic, and speed statistics models.

Chapter 5, Operations describes the continual operation requirements for the IMRCP system.

Chapter 6, Maintenance describes the maintenance required for the IMRCP system.

Chapter 7, References, lists the references of the IMRCP Administration/Installation Guide.

Appendix A, Log File Example, provides an example excerpt of the IMRCP system log.

Appendix B, Config.json Definitions, documents the names, types, and definitions of parameters in the IMRCP system configuration file.

Appendix C, DYNASMART-X Real-Time Dynamic Traffic Assignment System, details the installation of the DYNASMART-X components for the TrEPS model.

Appendix D, Glossary, lists the acronyms used throughout the document.

Chapter 2 Installation

Server Environment

The Integrated Model for Road Condition Prediction (IMRCP) system package contains all of the IMRCP system components other than Traffic Estimation and Prediction System (TrEPS). The IMRCP and TrEPS share underlying road network configuration information, but are implemented separately for each deployment area (for example, the Kansas City study area used for the initial deployment). Data exchange between the core IMRCP and TrEPS components consists of operational data files from the IMRCP sent to TrEPS and traffic estimation/prediction results from TrEPS read back into the IMRCP for the user interface and data archives. Figure 1 shows the deployment of the IMRCP system.

The IMRCP and TrEPS systems have run remotely "side by side" for the implementation and evaluation period in order to facilitate maintenance and updates by the respective support teams. Deployment of the TrEPS system within a co-located server environment is equally appropriate, but was not part of the demonstration deployment.

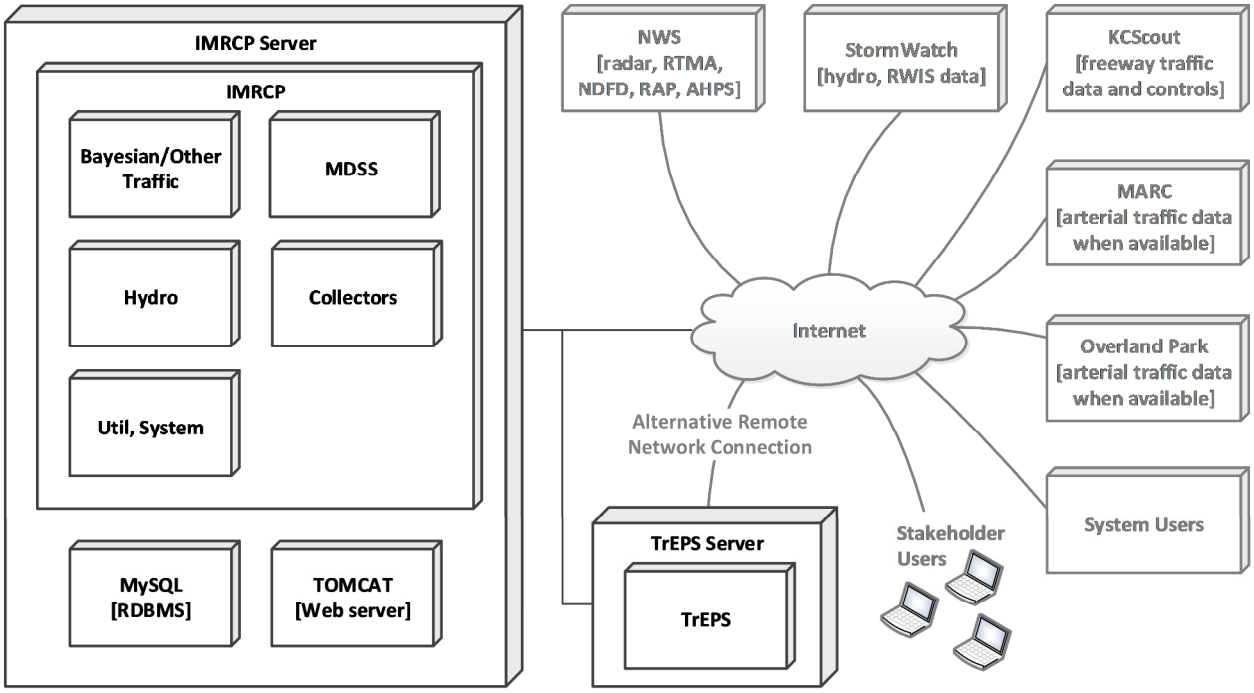

Figure 1. IMRCP Deployment Diagram

IMRCP Server

The application/web server includes 64 64-bit processors, 1 TB of memory, and 9 TB RAID5 solid state disk (SSD) storage. The system is run on Debian 64-bit Linux operating system with a 10 Mb symmetric internet connection. The database server is an open source MariaDB Server. Apache Tomcat 8 software is used for the application/web server. Java Runtime Environment 1.8 must be installed on the server.

The following local Java libraries are required for the installation of the IMRCP:

- Netcdf
- Database driver (in our case Mariadb)
- Metro
- Log4j2
- Apache Commons Compress
- Apache Commons Net
- Httpclient
- Bayes
- JDK 1.8
- Java EE Web 7 API Library
- Javax.json

TrEPS Server

The minimum server requirements for installing DYNASMART-X are:

- Platform: Windows XP and later versions.
- Memory: totally 8 GB RAM or above on one or multiple computers are recommended.
- Hard Drive Space: minimal 20 GB.
- Recommended display: Small fonts, 1024 x 768 screen resolution.

IMRCP Components

Installing the IMRCP system on a server involves multiple dependent steps:

1. First, a relational database management system (RDBMS) must be selected and installed. For the demonstration deployment, MariaDB was used. The process of installing a RDBMS will vary depending on which one is selected. To install the selected RDBMS, follow the documentation provided for that specific one. After installation and desired configuration of the database server is complete, start the database server and execute the create table scripts provided in SDK/deploy/db_table_scripts.txt.
2. A similar process needs to be carried out for a web application server. Apache Tomcat was selected for the demonstration deployment. Once the web application server is installed, move the SDK/deploy/IMRCP/ directory into its web applications directory.
3. Operating system privileges for files and directories need to be set to ensure a secure system. Tomcat documentation suggests not running a web application under the root user. Create a dedicated user for the web application process that has the minimum necessary permissions for the operating system.
4. Now that all of the components are installed, configuration files need to be completed or created. The main configuration file for IMRCP, config.json, by default is missing file paths for a lot of the components. These can be found in the file by searching for “<desired file path>” and “<desired temp file path>”. In the demonstration deployment, “/opt/imrcp/” was used for the desired file path, and “/dev/shm/imrcp/” was used for the desired temp file path. /dev/shm/ was used to prevent reading and writing to disk for frequently ran processes that used small files. Config.json also contains the components of IMRCP that are loaded upon system startup. These can be found under the “imrcp.system.Directory” section. By default, components that depend on a road network model are disabled. They are still listed but have a pound symbol (#) at the beginning of their name. Removing the pound symbol will enable the component. Before enabling these components, configuration files need to be created for the road network model and other component specific models. The components that are enabled by default are mainly weather and system related. The file path for config.json must be specified in the web.xml file in the

ImrcpDirectory servlet section. Similarly, the file path for the log file must be specified in the log4j2.properties file.

5. Another configuration item that needs to be set is the map tile service. Mapbox was the service used during the demonstration deployment. Once a map tile service has been selected, the URL for the background map and the road layer needs to be specified in SDK/deploy/IMRCP/ROOT/script/summary-map.js. Search for and replace “<background map layer url>” and “<road label layer url>” with the correct respective URL.

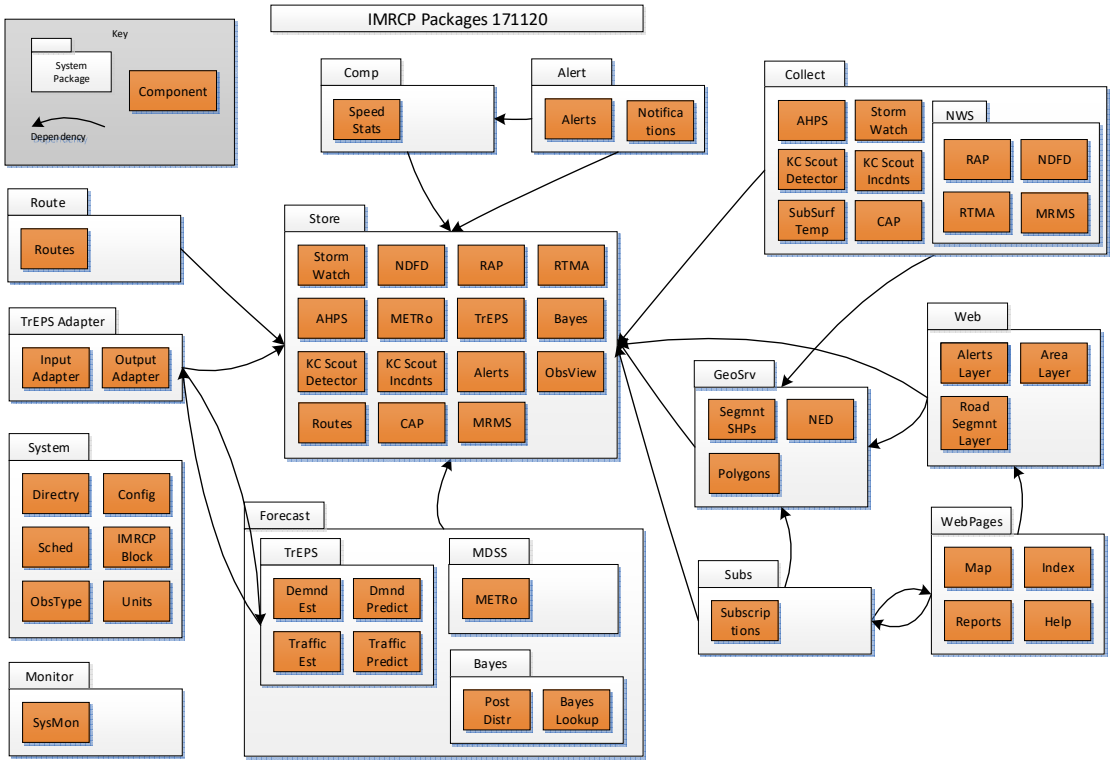

Figure 2. IMRCP Components

TrEPS Components

Installing DYNASMART-X on a plain machine requires several dependent steps. The overall installation flow chart is shown as in Figure 3.

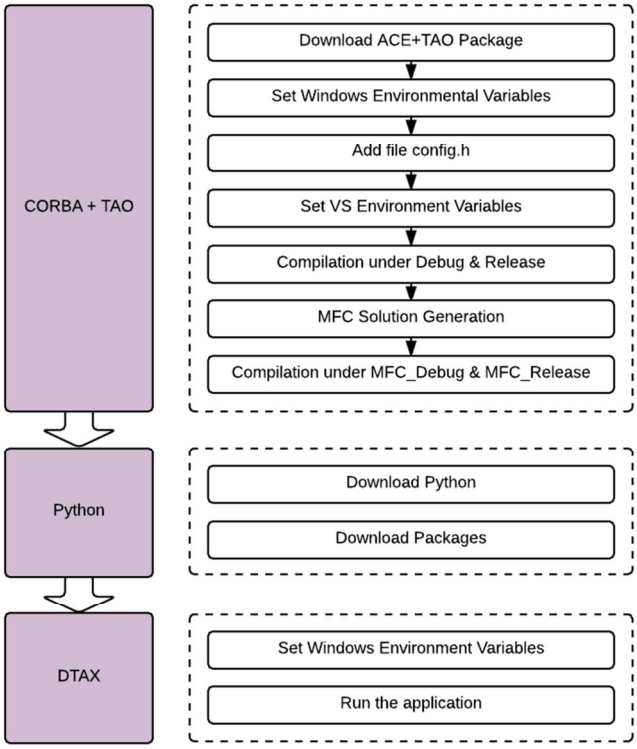

Figure 3. TrEPS Installation process flowchart

Installation of the TrEPS components is described in Appendix C, which is excerpted from the DYNASMART-X Real-Time Dynamic Traffic Assignment System Manual. It assumes Visual Studio and Fortran Complier have already been installed and focuses on installation of the TAO and DTAX packages. Other assumptions of this manual are listed below:

- The operating system is Windows 7.
- Machine is 64-bit.
- Visual Studio version is 2005 (recommended).
- Intel Fortran Complier version is 10.1.
- User is logged on as Administrator.
- Python 2.7 is installed on the Windows server.

Chapter 3 Application Configuration

Configuration Files

Config.json

All of the configurable parameters for ImrcpBlocks and other system components are contained in config.json. The file is written in JSON format. It consists of an array of objects with each object having a key and a value. The key is an ImrcpBlock fully-qualified Java name, instance name, or, in the case of a non-ImrcpBlock, a string. The value is another object that contains the configuration key/value pairs for that key. The values of a configuration pair can be an integer, a string, or a string array. The configuration object for an ImrcpBlock first searches the “imrcp.ImrcpBlock” section for configured values. Next it will search the section named by the fully-qualified java name, which will override any values from the “imrcp.ImrcpBlock” section. Finally, it will search the section named by its instance name, which will override any values from the fully-qualified java name section. Configuration sections for non ImrcpBlock must be referenced by name in the code. Any reference to a string obs type in the tables below is the 6-character-or-less name defined in the ObsType class that is used to create the integer obs type. The config.json definitions can be found in 0.

Context.xml

This file contains the jdbc resource for the web application. Its path is application root/META-INF/context.xml.

The database pool connection is defined here.

Web.xml

This file contains configuration parameters for the Tomcat web application. Its path is application root/WEB-INF/web.xml. It contains the IMRCPblock for directory which starts up when Tomcat starts. It is responsible for reading the tomcat json and starting up the other Imrcpblocks.

Polygons (Localization Required)

Polygons contains the geometric definitions for FIPS and UGC geocodes. These definitions are used for the National Weather Service (NWS) Alerts area data layer. For the Kansas City study area, only the FIPS and UGC geocodes in and near the area are contained in the configuration file. This file would require changes for other IMRCP locations.

Segments (Localization Required)

The segments configuration file contains segment definitions based off of the road network model provided by Northwestern University Transportation Center (NUTC) for the Kansas City study area. Changes must be made to this configuration file for other IMRCP locations.

log4j2.properties

This configuration file contains the configuration for the logger log4j2.

Detector_Link_Relationships (Localization Required)

The Kansas City study area Detector_Link_Relationship file contains the mappings for the KC Scout detectors to links in the road network model. This configuration file contains data specific to the IMRCP location.

Highway Segments (Localization Required)

This configuration file contains a list of the segment IDs that are classified as highways and is specific to the location of the of the IMRCP deployment.

Bridge Coordinates (Localization Required)

This configuration file contains the coordinates of bridges in the study area. This file is used to split links to create segments, so that segments may be classified as bridges or non-bridges. Data in this file is unique to the IMRCP deployment location.

Routes (Localization Required)

The Routes configuration file contains lists of nodes that create routes. These lists are used to generate route travel times and display routes on the map. Because nodes and rountes are specific to the location of the IMRCP deployment, this file must be modified for each location case.

Unit Conversions

This configuration file contains values to create unit conversion objects in the system.

Source Units (Localization Required)

The Source Units configuration file indicates the units used in source files and specifies what units to use for each observation type. Because each deployment location requires a different set of sources, this conifiguration file must be modified for each of the locations.

AHPS (Localization Required)

The AHPS configuration file uses the inundation maps provided by NWS to map road flood depths to study area segments using their latitudes and longitudes. Different AHPS stations must be used in deployment locations, so the AHPS configuration file must be modified to match the inundation mappings of those locations.

Chapter 4 Model Configuration

Road Network Model

The road network model is generated from SHP files produced by NUTC. The SHP files include latitude and longitude coordinates (WGS-84 datum) of links with a specified geometry between two nodes. The links include both the highways and arterials in the study area.

At least one segment is generated per link from the SHP files for the road network model. Links may be split into two or more segments if the link contains a bridge. Bridge segment endpoints can be identified from maps with satellite image backgrounds (e.g., OpenStreetMaps (OSM)) and are contained in a Bridge Coordinates configuration file. The system uses a function to iterate through each point of geometry on a link until the bridge endpoints in the Bridge Coordinates configuration file are within the specified tolerance of a point on that link. This point is where the link is split to form segments.

Links, segments, and nodes are stored in the database with unique system ID's consistent with the IMRCP ID format. Pavement and traffic observations are mapped to specific segments. The road network model is displayed on the map user interface where users can select segments to view observations.

Data from the National Elevation Database[1] (NED) is used to look up the elevation at the midpoint of each segment. This elevation is stored with the segment as the segment's elevation. The NED is also used to calculate the grade of each link. To calculate the grade of the segment, the elevation of the start point is subtracted from the elevation of the endpoint and divided by the length of the link. Figure 4 shows this calculation.

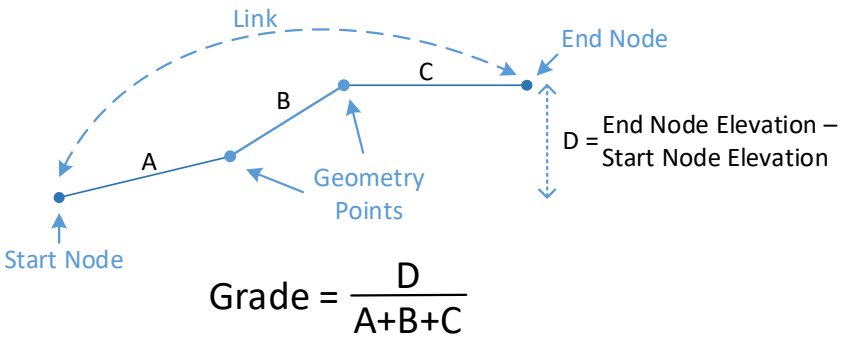

Figure 4. Grade Calculation

Detectors

KC Scout Detectors provide real-time and archived speed, volume, and occupancy data to the IMRCP system. These data are used in the calibration of the traffic demand model. KC Scout provided a CSV file including the geometric coordinates, name, external ID (used in real-time feed), internal ID (used in

[1] United States Geological Service, "The National Map Elevation" web page. Available at: https://nationalmap.gov/elevation.html.

archive data), and location reference for each detector in the Kansas City area. A new file titled Detector_Link_Relationships.csv file was created based on the CSV file provided by KC Scout.

Detector_Link_Relationship.csv lists all of the detectors in the IMRCP study area. The geometric coordinates, name, real-time ID (KC Scout external ID), and archive ID (KC Scout internal ID) for each of these detectors are included in the Detector_Link_Relationship.csv file. Columns have been added to the file to indicate detector association with a ramp or a metered ramp. Each detector was manually mapped to a link in the model. The I-node, J-node, and link ID from the NUTC model for each link are also listed in the Detector_Link_Relationship.csv file.

The Detector_Link_Relationship.csv file is converted to the detector_mapping_to_link.txt by the system. This file includes geometric coordinates, name, real-time ID, archive ID, and link mapping for each detector in the study area. Detector_mapping_to_link.txt is used by the TrEPS system to map each detector to a link in the model.

Detector_Link_Relationships.csv is used to determine which detectors to collect data from the real-time feed. The data are collected from KC Scout's TransSuite real-time data portal as an XML file every minute. The data collected from the real-time data portal are from the previous 30 seconds. The real-time data is fed to TrEPS using detector.dat. Detector_Link_Relationships.csv is used to create a list for detector.dat and the list is filled in based on what was collected from the real-time feed.

The detector.dat file implements the real-time detector data interface from the core IMRCP system to the TrEPS/DYNASMART model. The real-time file can be found and inspected at https://imrcp.data-env.com/detector.dat. Detector.dat is updated every minute and includes the archive ID, timestamp of each observation, volume, speed, occupancy, and link ID for each detector for each of the previous 15 minutes. For the Kansas City demonstration, the total volume per minute is calculated by summing the volume per lane for each detector from the real-time 30-second feed and multiplying it by two. The average speed and occupancy per detector per minute are calculated by taking the average of the speed and occupancy for each lane for each detector from the real-time feed. If a detector is not in service for one of the minutes in the file, detector.dat reports a speed, volume, and occupancy of -100 for that minute.

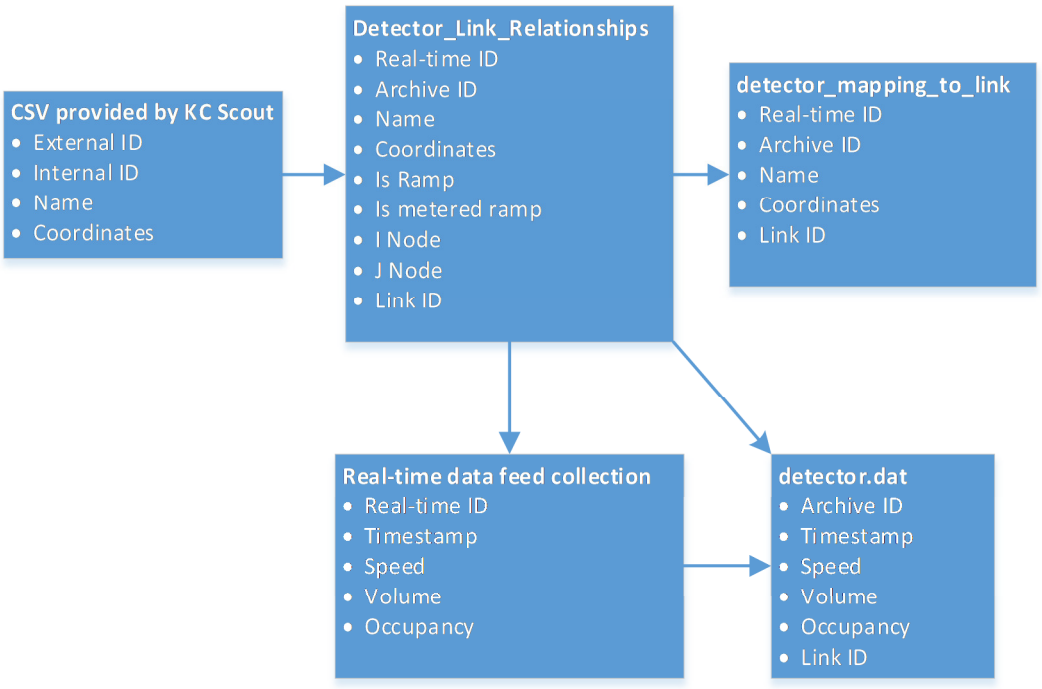

Figure 5. Detector Configuration Data Relationships

The detector_mapping_to_link.txt file describes the sensor/detector locations in the network. Six link parameters are used to represent the detector mapping information for real-time observation data. The six link parameters are the detector id, longitude, latitude, link id, upstream node, and downstream node.

TrEPS Traffic Demand Model

Information on the configuration of TrEPS can be found in Appendix C.

Road Weather Model

The Model of the Environment and Temperature of Roads (METRo) is run on all segments in the study area to provide estimates and forecasts of road weather conditions. As described in the IMRCP Model Analysis, METRo is a standard pavement thermal modeling tool developed by the Canadian Meteorological Center of Environment Canada as part of its road weather forecasting suite.[2] METRo computes the road temperature, subsurface temperature, pavement state, liquid depth, and snow/ice depth.

[2] Crevier, L.-P., and Y. Delage. *METRo: A New Model for Road-Condition Forecasting in Canada. Journal of Applied Meteorology Vol. 40*, November 2001, pages 2026-2037. Downloaded at: http://journals.ametsoc.org/doi/pdf/10.1175/1520-0450%282001%29040%3C2026%3AMANMFR%3E2.0.CO%3B2.

METRo needs the recent history of the observations, the station configurations, and weather forecast data. External software is used to collect the data needed by METRo to interface with it. Observations are collected from the sources in Table 1.

Table 1. Road Weather Model Observation Data Sources

Observation Type	Source
air temperature	RTMA
wind speed	RTMA
dew point temperature	RTMA
road condition	previous METRo file if available, otherwise set to a configured value
road temperature	previous METRo file if available, otherwise set to a configured value
subsurface temperature	previous METRo file if available, otherwise taken from an RWIS station

METRo is configured so that the first hour of the forecast is one hour before the current time. METRo requires at least 20 minutes of data to calibrate its calculations. The one-hour time shift is necessary to accommodate the minimum amount of data required (the first interval of atmospheric forecast data greater than 20 minutes) and to synchronize its results for presentation. The forecast data for METRo is collected from the data sources in Table 2.

Table 2. Road Weather Model Forecast Data Sources

Observation Type	Hour 1 Source	Hours 2-10 Source
precipitation amount	MRMS	RAP
precipitation type	Inferred based on air temperature	RAP
rain reservoir	previous METRo file	previous METRo file
snow reservoir	previous METRo file	previous METRo file
air temperature	RTMA	NDFD
dew point temperature	RTMA	NDFD
wind speed	RTMA	NDFD
cloud coverage	RTMA	NDFD
surface pressure	RTMA	RAP

METRo refers to each evaluated location as a “station”. The following “station” variables are collected from the database:

- If the segment is a bridge.
- Latitude.
- Longitude.
- Treatment type.

These data are sent to the METRo library functions (adapted from an NCAR method) using Java Native Interface (JNI). All forecast and observation arrays get interpolated from every one hour to every 30 seconds. Then, the solar and infrared fluxes and absolute humidity are calculated. The METRo model is called using all of the data. Outputs return forecast observations for road condition, road temperature, subsurface temperature, snow/ice depth, and liquid depth.

For the current Kansas City demonstration deployment, METRo is run every 20 minutes and is executed at each segment/site. METRo is configured for the IMRCP to consider rain reservoir above 0.2 mm to cause wet pavement.

Hydrological Model

The hydrological model for IMRCP uses data collected from three Advanced Hydrological Prediction System (AHPS) stations in the study area – Indian Creek at Overland Park, Kansas; Indian Creek at State Line Road, Kansas; and Tomahawk Creek at Roe Avenue, Kansas. The new flood depth is collected when it becomes available at any of the stations.

A CSV file was created using the inundation mapping available for each of the three stations in the study area on the AHPS website.[3] The CSV file lists coordinates of roads that are covered at each inundation level for each station and the flood depth on that road according the the inundation map. When a new flood depth value is collected from AHPS, the IMRCP checks the value against the list of flood stage values in the CSV file. If the file indicates that any roads are covered at that flood depth, the system uses a snap function to snap the coordinates listed in the CSV file to the nearest link. An observation is then created for that link based on the flood depth value in the file. The observation will remain in effect until a new flood depth is collected from AHPS or for the next one hour.

Bayes Traffic Model

A Bayesian network traffic model is used to estimate traffic characteristics on highway segments based on their operational and weather characteristics. As shown in Figure 6, the key components are the data store of historical and current traffic states, the Estimator (which builds a statistical record of traffic states), and a Predictor (which evaluates an anticipated state against the statistical record). The Estimator and Predictor use a Bayesian network of state parameters to characterize the state of the traffic network.

[3] National Oceanic and Atmospheric Administration, National Weather Service, "U.S. Water Map" web page. Available at: https://water.weather.gov/ahps/.

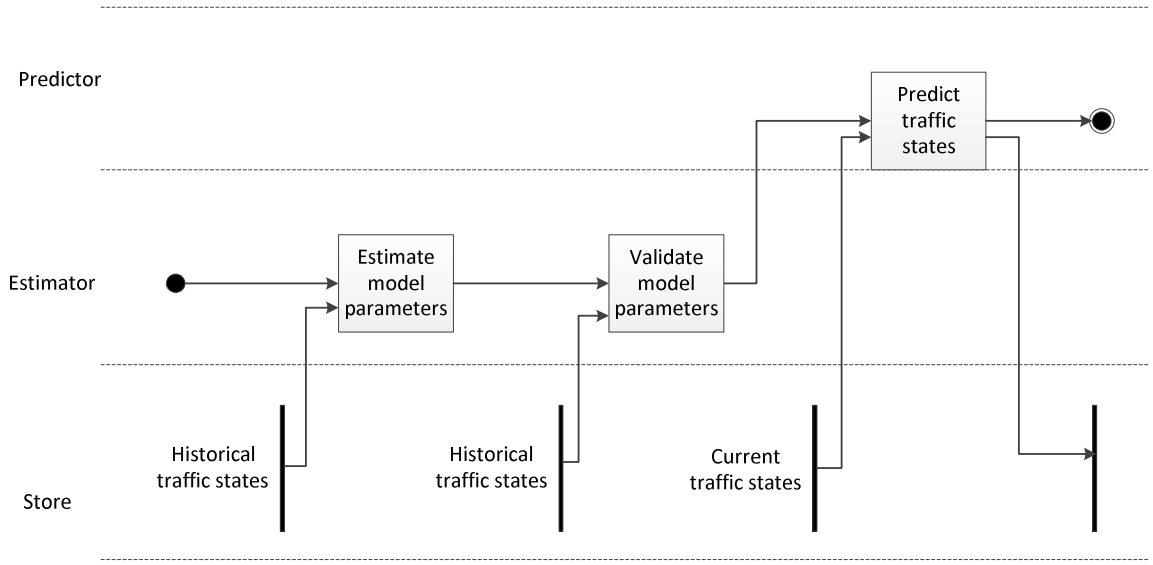

Figure 6. Bayesian Network Process

The Bayesian Estimator model starts with the Bayesian network structure. The Bayesian Estimator uses two years of historical traffic data from the study area. Then based on the weather, roadwork, traffic and other data inputs, the estimator computes and estimates the distributions of all variables (nodes) in that Bayesian network.

Based on the updated Bayesian network model from the Estimator, the Predictor uses data on current link traffic states (e.g., direction, weather, and current time) and other system variables, such as work zones and incidents, from the data Store as model inputs. Using the estimated distributions of system variables, the Bayesian Predictor predicts the future network states of traffic flow, speed, and occupancy. The input variables for the Bayesian Predictor are listed below in Table 3.

This model is implemented in the IMRCP as a lookup of the Bayesian Predictor state results. An executable for the Bayesian Predictor is used to determine the network state. The current state, characterized by its set of system variables is checked against the available lookup values. When a new set is found, the executable is run for those inputs and stored in the lookup table. If the executable has been run previously for a set of variables, the values determined by that executable are used.

Variables used in the IMRCP Bayesian network model are presented in Table 3. A total of 14 variables are categorized into three groups. In each group, each variable does not have a relationship with other variables in the same group (i.e., each variable does not affect other variables in the same group). The relationships of the three groups are listed below:

- Variables in Group 1 (Network Environment) can directly affect variables in Group 2 (External Event) and Group 3 (Traffic Condition), but not the other way around (e.g., weather, such as snow, can cause an incident, but an incident cannot cause a weather event).
- Variables in Group 2 (External Event) can directly affect variables in Group 3 (Traffic Condition).
- Variables in Group 3 are different measures of the traffic condition on a particular link. They are independent from each other.

Table 3. Bayesian Network Variable

Node Group	Variable	States	State Definitions
Group 1: Network Environment	Direction	-Eastbound -Southbound -Westbound -Northbound	
	Weather	-Clear -Light rain, clear visibility -Light rain, reduced visibility -Light rain, low visibility -Moderate rain, clear visibility -Moderate rain, reduced visibility -Moderate rain, low visibility -Heavy rain, reduced visibility -Heavy rain, low visibility -Light snow, clear visibility -Moderate snow, reduced visibility -Heavy snow, reduced visibility -Heavy snow, low visibility	00mm/h; visibility >3300ft < 2.5mm/h; > 3300ft < 2.5mm/h; 330 - 3300ft < 2.5mm/h; < 330ft 2.5 - 7.6mm/h; > 3300ft 2.5 - 7.6mm/h; 330 - 3300ft 2.5 - 7.6mm/h; < 330ft ≥ 7.6mm/h; 330 - 3300ft ≥ 7.6mm/h; < 330ft ; >3300ft* ; 1650 - 3300ft* ; 330 - 1650ft* ; < 330 ft*
	DayOfWeek	-Weekend -Weekday	Saturday, Sunday Monday - Friday
	TimeOfDay	-Morning -AM peak -Off-peak -PM peak -Night	1AM - 6AM (5 hrs) 6AM - 10 AM (4hrs) 10AM - 4PM (6hrs) 4PM - 8PM (4hrs) 8PM - 1AM (5 hrs)
Group 2: External Event	IncidentDownstream	-No incident -Incident	
	IncidentOnLink	-No incident -Incident	
	IncidentUpstream	-No incident -Incident	
	Workzone	-No work zone -Workzone	
	RampMetering	-No ramp metering -Ramp metering	
	SpecialEvents**	-Special event -No special event	
Group 3: Traffic Condition	Flow (veh/h/ln)	-Very low -Low -High -Very high	< 0.25*** 0.25 - 0.5 0.5 - 0.75 ≥ 0.75
	Speed (mph)	-Very low -Low -High -Very high	< 0.25*** 0.25 - 0.5 0.5 - 0.75 ≥ 0.75
	Occupancy (%)	-Very low -Low -High -Very high	< 0.25*** 0.25 - 0.5 0.5 - 0.75 ≥ 0.75

* Snowfall's intensity is determined by visibility. https://en.wikipedia.org/wiki/Snow.

** This is one potential way to accommodate "special events" in the Bayesian Network model. The "special event" here is nominal. It can represent any special event, such as football or hockey games. The special event will be defined when real world data are used to train the BN model. Any special event in Bayesian model should be considered on a case-by-case basis.

*** Based on normalized parameter values, which range between 0 and 1

Configuration File

BayesPredictModel.net

This file contains the trained Bayesian Network for the road network (e.g., the Kansas City demonstration study area).

Speed Statistics Model

The speed statistics model calculates the expected normal speed for each link associated with a detector. The calculation is based on historical detector data; two years of data from KC Scout were used for the demonstration study area. The average speed over the specified day and time is used to determine off-normal conditions, configured for the Kansas City demonstration to 70% of the expected normal speed. Determination of an off-normal condition triggers alerts and notifications on the user interface. The current implementation is specific to the Kansas City demonstration area based on archived data from the KC Scout ATMS.

Chapter 5 Operations

Log File

The system is run upon startup of the Tomcat server. The logging system (log4j v2) puts component-specific messages in the log file. The log file lists information on the IMRCP components as well as errors that occur in the system. Each line in the log file includes an indicator word:

- INFO: information about a system component activity.
- ERROR: information about an error that has occurred within a component.
- DEBUG: information used to help debug the system.

Each line also lists the date and time of the message, the component, and the message. An example of a portion of the log file can be seen in 0.

System Startup and Shutdown

The system starts up and shuts down through the selected web application server. For Tomcat, the commands are:

- *tomcat_directory*/bin/catalina.sh start
- *tomcat_directory*/bin/catalina.sh stop

Chapter 6 Maintenance

Backup

System backup is run externally to the system. It is recommended that the backup be performed weekly. Older files may be archived to free up space for the system. Archiving the files removes them from the presentation and reports.

User Accounts

User accounts are configured in tomcat-users.xml. Adding, removing, and modifying usernames and passwords can be done in this file. Users can also be classified as “imrcp-user” or “imrcp-admin”. Users with admin privileges have access to additional system functions specified in the system. In order for changes to tomcat-users.xml to take effect, the system must be restarted. An example of the line that must be added to tomcat-users.xml to add a user to the system is listed below:

- <user username="nameu" password="pa55word" roles="imrcp-user"/>

Chapter 7 References

Leidos, *Integrated Modeling for Road Condition Prediction Model Analysis* (unpublished working paper developed under Contract DTFH61-12-D-00050, Integrated Modeling for Road Condition Prediction, May 10, 2015.

Leidos, *Integrated Modeling for Road Condition Prediction Concept of Operations* (unpublished working paper developed under Contract DTFH61-12-D-00050, Integrated Modeling for Road Condition Prediction, November 25, 2015).

Leidos, *Integrated Modeling for Road Condition Prediction System Requirements* (unpublished working paper developed under Contract DTFH61-12-D-00050, Integrated Modeling for Road Condition Prediction, January 25, 2016).

Leidos, *Integrated Modeling for Road Condition Prediction System Design Description* (unpublished working paper developed under Contract DTFH61-12-D-00045, Integrated Modeling for Road Condition Prediction-Phase 2, December 28, 2016).

Appendix A. Log File Example

```
[INFO  2017-08-15 18:37:59.780 [RadarPrecipStore] - Finished loading
/opt/imrcp-prod/mrms/precip/201708/20170815/precip_010_20170815_1646_000.grb2
[INFO  2017-08-15 18:37:59.780 [RadarPrecipStore] - Loading /opt/imrcp-
prod/mrms/precip/201708/20170815/precip_010_20170815_1648_000.grb2 into
memory: Deque
[INFO  2017-08-15 18:37:59.799 [RAPStore] - Finished loading /opt/imrcp-
prod/rap/201708/20170815/rap_130_20170815_1500_002.grb2
[INFO  2017-08-15 18:37:59.800 [RAPStore] - Loading /opt/imrcp-
prod/rap/201708/20170815/rap_130_20170815_1500_003.grb2 into memory: Deque
[INFO  2017-08-15 18:37:59.994 [RAPStore] - Finished loading /opt/imrcp-
prod/rap/201708/20170815/rap_130_20170815_1500_003.grb2
[INFO  2017-08-15 18:37:59.994 [RAPStore] - Loading /opt/imrcp-
prod/rap/201708/20170815/rap_130_20170815_1500_004.grb2 into memory: Deque
[DEBUG 2017-08-15 18:38:00.001 [KCScoutDetectors] - Starting getDetectors()
[INFO  2017-08-15 18:38:00.188 [KCScoutIncidents] - KCScoutIncidents notified
KCScoutIncidentsStore: file download
[INFO  2017-08-15 18:38:00.216 [KCScoutIncidentsStore] -
KCScoutIncidentsStore notified RealTimeIncident: new data
[INFO  2017-08-15 18:38:00.217 [KCScoutIncidentsStore] -
KCScoutIncidentsStore notified RealTimeWorkzone: new data
[INFO  2017-08-15 18:38:00.218 [KCScoutIncidentsStore] -
KCScoutIncidentsStore notified IncidentNotifications: new data
[INFO  2017-08-15 18:38:00.246 [RAPStore] - Finished loading /opt/imrcp-
prod/rap/201708/20170815/rap_130_20170815_1500_004.grb2
[INFO  2017-08-15 18:38:00.247 [RAPStore] - Loading /opt/imrcp-
prod/rap/201708/20170815/rap_130_20170815_1500_005.grb2 into memory: Deque
[INFO  2017-08-15 18:38:00.254 [KCScoutIncidentsStore] - Loading /opt/imrcp-
prod/incident/201708/20170815_eventsList.csv into memory: Lru
[INFO  2017-08-15 18:38:00.259 [KCScoutIncidentsStore] - Finished loading
/opt/imrcp-prod/incident/201708/20170815_eventsList.csv
[INFO  2017-08-15 18:38:00.342 [IncidentNotifications] -
IncidentNotifications notified IncidentNotificationsStore: file download
[INFO  2017-08-15 18:38:00.342 [IncidentNotificationsStore] - Loading
/dev/shm/imrcp-prod/incident_notifications.csv into memory: Deque
[INFO  2017-08-15 18:38:00.343 [IncidentNotificationsStore] - Loading
/opt/imrcp-prod/notification/incident/201708/notification_20170815.csv into
memory: Lru
[INFO  2017-08-15 18:38:00.351 [IncidentNotificationsStore] - Finished
loading /opt/imrcp-
prod/notification/incident/201708/notification_20170815.csv
[INFO  2017-08-15 18:38:00.601 [RAPStore] - Finished loading /opt/imrcp-
prod/rap/201708/20170815/rap_130_20170815_1500_005.grb2
[INFO  2017-08-15 18:38:00.602 [RAPStore] - Loading /opt/imrcp-
prod/rap/201708/20170815/rap_130_20170815_1500_006.grb2 into memory: Deque
[ERROR 2017-08-15 18:38:00.728 [TrafficAlertsStore] - File does not exist:
/opt/imrcp-prod/alert/traffic/201708/alert_20170815.csv
[ERROR 2017-08-15 18:38:00.728 [AHPSAlertsStore] - File does not exist:
/opt/imrcp-prod/alert/ahps/201708/alert_20170815.csv
[ERROR 2017-08-15 18:38:00.728 [SpeedStatsAlertsStore] - File does not exist:
/opt/imrcp-prod/alert/speed/201708/alert_20170815.csv
```

```
[INFO  2017-08-15 18:38:00.932 [RAPStore] - Finished loading /opt/imrcp-
prod/rap/201708/20170815/rap_130_20170815_1500_006.grb2
[INFO  2017-08-15 18:38:00.933 [RAPStore] - Loading /opt/imrcp-
prod/rap/201708/20170815/rap_130_20170815_1500_007.grb2 into memory: Deque
```

Appendix B. Config.json Definitions

imrcp.system.Directory

Key Name	Type	Purpose
classes	string array	Tells the system what classes to load at startup. The array consists of string pairs. The first string is the fully qualified Java name of the class, and the second string is the instance name the system will use to identify this instance of the class.

imrcp.ImrcpBlock

All configuration objects get these values; they are the same for each block. The values can be overridden in block specific configuration.

Key Name	Type	Purpose
box	string array	Bounding box of the study area. The array consists of groups of 4 strings that represent rectangles that make up the study area. The format of the 4 strings is left, bottom, right, top each one written as integer degrees scaled to 7 decimal places.
retryint	int	Used primarily for collectors, represents the time, in milliseconds, to wait before retrying to download a file that was not available.
retrymax	int	Used primarily for collectors, represents the maximum times a collector retries to download a missed file.
timeout	string	The time, in milliseconds, this block waits for other blocks it is subscribed to finish their start method before this calls its start method.

imrcp.store.NDFDStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dir	string	String used to create the SimpleDateFormat that uses the time to create the file structure.
subobs	string array	Array of obs type strings that this store produces.

imrcp.store.NDFDTdStore

Key Name	Type	Purpose
filepattern	string	Regular expression used to detect the correct files in the directory.
obsid	string array	Array of internal obs type strings that map to the observations in the netCDF files. Need to be in the same order as the “obs” configured array.
obs	string array	Array of observation names that are used from the netCDF files. Need to be in the same order as the “obsid” configured array to map the observations correctly.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
limit	int	The maximum number of files that can be loaded into the current files cache in memory for this block.
delay	int	The time, in milliseconds, to takes for a file to be valid after it has

		been collected.
range	int	The length of time, in milliseconds, that a file is valid.
hrz	string	The name of the horizontal axis in the netCDF file.
vrt	string	The name of the vertical axis in the netCDF file.
time	string	The name of the time axis in the netCDF file.
keeptime	Int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.
lrulim	int	The maximum number of files that can be loaded into the lru in memory for this block.
freq	int	How often a file is collected, in milliseconds, for this block.

imrcp.store.NDFDTempStore

Key Name	Type	Purpose
filepattern	string	Regular expression used to detect the correct files in the directory.
obsid	string array	Array of internal obs type strings that map to the observations in the netCDF files. Need to be in the same order as the "obs" configured array.
obs	string array	Array of observation names that are used from the netCDF files. Need to be in the same order as the "obsid" configured array to map the observations correctly.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
limit	int	The maximum number of files that can be loaded into the current files cache in memory for this block.
delay	int	The time, in milliseconds, to takes for a file to be valid after it has been collected.
range	int	The length of time, in milliseconds, that a file is valid.
hrz	string	The name of the horizontal axis in the netCDF file.
vrt	string	The name of the vertical axis in the netCDF file.
time	string	The name of the time axis in the netCDF file.
keeptime	int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.
lrulim	int	The maximum number of files that can be loaded into the lru in memory for this block.
freq	int	How often a file is collected, in milliseconds, for this block.

imrcp.store.NDFWspdStore

Key Name	Type	Purpose
filepattern	string	Regular expression used to detect the correct files in the directory
obsid	string array	Array of internal obs type strings that map to the observations in the netCDF files. Need to be in the same order as the "obs" configured array.
obs	string array	Array of observation names that are used from the netCDF files. Need to be in the same order as the "obsid" configured array to map the observations correctly.

dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
limit	int	The maximum number of files that can be loaded into the current files cache in memory for this block.
delay	int	The time, in milliseconds, to takes for a file to be valid after it has been collected.
range	int	The length of time, in milliseconds, that a file is valid.
hrz	string	The name of the horizontal axis in the netCDF file.
vrt	string	The name of the vertical axis in the netCDF file.
time	string	The name of the time axis in the netCDF file.
keeptime	Int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.
lrulim	int	The maximum number of files that can be loaded into the lru in memory for this block.
freq	int	How often a file is collected, in milliseconds, for this block.

imrcp.store.NDFDSkyStore

Key Name	Type	Purpose
filepattern	string	Regular expression used to detect the correct files in the directory
obsid	string array	Array of internal obs type strings that map to the observations in the netCDF files. Need to be in the same order as the “obs” configured array.
obs	string array	Array of observation names that are used from the netCDF files. Need to be in the same order as the “obsid” configured array to map the observations correctly.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
limit	int	The maximum number of files that can be loaded into the current files cache in memory for this block.
delay	int	The time, in milliseconds, to takes for a file to be valid after it has been collected.
range	int	The length of time, in milliseconds, that a file is valid.
hrz	string	The name of the horizontal axis in the netCDF file.
vrt	string	The name of the vertical axis in the netCDF file.
time	string	The name of the time axis in the netCDF file.
keeptime	Int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.
lrulim	int	The maximum number of files that can be loaded into the lru in memory for this block.
freq	int	How often a file is collected, in milliseconds, for this block.

imrcp.store.RTMAStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.

limit	int	The maximum number of files that can be loaded into the current files cache in memory for this block.
filepattern	string	Regular expression used to detect the correct files in the directory.
delay	int	The time, in milliseconds, to takes for a file to be valid after it has been collected.
range	int	The length of time, in milliseconds, that a file is valid.
obsid	string array	Array of internal obs type strings that map to the observations in the netCDF files. Need to be in the same order as the "obs" configured array.
obs	string array	Array of observation names that are used from the netCDF files. Need to be in the same order as the "obsid" configured array to map the observations correctly.
hrz	string	The name of the horizontal axis in the netCDF file.
vrt	string	The name of the vertical axis in the netCDF file.
time	string	The name of the time axis in the netCDF file.
keeptime	Int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
lrulim	int	The maximum number of files that can be loaded into the lru in memory for this block.
subobs	string array	Array of obs type strings that this store produces.
freq	int	How often a file is collected, in milliseconds, for this block.

imrcp.store.RAPStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
limit	int	The maximum number of files that can be loaded into the current files cache in memory for this block.
filepattern	string	Regular expression used to detect the correct files in the directory.
delay	int	The time, in milliseconds, to takes for a file to be valid after it has been collected.
range	int	The length of time, in milliseconds, that a file is valid.
hrz	string	The name of the horizontal axis in the netCDF file.
vrt	string	The name of the vertical axis in the netCDF file.
time	string	The name of the time axis in the netCDF file.
keeptime	int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
lrulim	int	The maximum number of files that can be loaded into the lru in memory for this block.
obs	string array	Array of observation names that are used from the netCDF files. Need to be in the same order as the "obsid" configured array to

		map the observations correctly.
obsid	string array	Array of internal obs type strings that map to the observations in the netCDF files. Need to be in the same order as the "obs" configured array.
subobs	string array	Array of obs type strings that this store produces.
freq	int	How often a file is collected, in milliseconds, for this block.
files	int	The number of files the RAP collector is downloading.

RapNcf

This section is used by the RapNcfWrapper class.

Key Name	Type	Purpose
lrain	string	The threshold for light rain.
mrain	string	The threshold for medium rain.
lsnow	string	The threshold for light snow.
msnow	string	The threshold for medium snow.

imrcp.store.NcfWrapper

Key Name	Type	Purpose
fcst	int	The time, in milliseconds, that a forecast is valid for this time of NcfWrapper.

imrcp.store.RadarNcfWrapper.mrms

Key Name	Type	Purpose
fcst	int	The time, in milliseconds, that a forecast is valid for this time of NcfWrapper.

imrcp.store.MetroStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
limit	int	The maximum number of files that can be loaded into the current files cache in memory for this block.
filepattern	string	Regular expression used to detect the correct files in the directory.
delay	int	The time, in milliseconds, to takes for a file to be valid after it has been collected.
range	int	The length of time, in milliseconds, that a file is valid.
hrz	string	The name of the horizontal axis in the netCDF file.
vrt	string	The name of the vertical axis in the netCDF file.
time	string	The name of the time axis in the netCDF file.
keeptime	Int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
lrulim	int	The maximum number of files that can be loaded into the lru in memory for this block.
obs	string array	Array of observation names that are used from the netCDF files. Need to be in the same order as the "obsid" configured array to

		map the observations correctly.
obsid	string array	Array of internal obs type strings that map to the observations in the netCDF files. Need to be in the same order as the "obs" configured array.
subobs	string array	Array of obs type strings that this store produces.
freq	int	How often a file is collected, in milliseconds, for this block.

imrcp.store.RadarStore

These values are shared by any instance of RadarStore

Key Name	Type	Purpose
limit	int	The maximum number of files that can be loaded into the current files cache in memory for this block.
delay	int	The time, in milliseconds, to takes for a file to be valid after it has been collected.
range	int	The length of time, in milliseconds, that a file is valid.
hrz	string	The name of the horizontal axis in the netCDF file.
vrt	string	The name of the vertical axis in the netCDF file.
time	string	The name of the time axis in the netCDF file.
keeptime	Int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.
lrulim	int	The maximum number of files that can be loaded into the lru in memory for this block.
freq	int	How often a file is collected, in milliseconds, for this block.

RadarStore

These values are specific to the reflectivity RadarStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dir	string	String used to create the SimpleDateFormat that is used to generate directory names.
filepattern	string	Regular expression used to detect the correct files in the directory
obsid	string array	Array of internal obs type strings that map to the observations in the netCDF files. Need to be in the same order as the "obs" configured array.
obs	string array	Array of observation names that are used from the netCDF files. Need to be in the same order as the "obsid" configured array to map the observations correctly.
subobs	string array	Array of obs type strings that this store produces.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.

RadarPrecipStore

These values are specific to the precipitation RadarStore

Key Name	Type	Purpose

attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dir	string	String used to create the SimpleDateFormat that is used to generate directory names.
filepattern	string	Regular expression used to detect the correct files in the directory.
obsid	string array	Array of internal obs type strings that map to the observations in the netCDF files. Need to be in the same order as the "obs" configured array.
obs	string array	Array of observation names that are used from the netCDF files. Need to be in the same order as the "obsid" configured array to map the observations correctly.
subobs	string array	Array of obs type strings that this store produces.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
limit	int	The maximum number of files that can be loaded into the current files cache in memory for this block.
keeptime	int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.

imrcp.collect.NDFD

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
offset	int	The offset in seconds used in scheduling this block's regularly executed task.
period	int	How often this block regularly executes its task in seconds.

imrcp.collect.RTMA

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
url	string	The url used to download files.
dir	string	The base directory for saving files.
time	int	The amount of time in seconds to go in the past to initially download files.
offset	int	The offset in seconds used in scheduling this block's regularly executed task.
period	int	How often this block regularly executes its task in seconds.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.

imrcp.collect.RAP

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
url	string	The url used to download files.
dir	string	The base directory for saving files.
time	int	The amount of time in seconds to go in the past to initially download files.
offset	int	The offset in seconds used in scheduling this block's regularly

		executed task.
period	int	How often this block regularly executes its task in seconds.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
test	string	Must be “True” or “False”, if “True” the block will run in test mode.

imrcp.collect.Radar

These values are shared for any Radar collector.

Key Name	Type	Purpose
time	int	The amount of time in seconds to go in the past to initially download files.
offset	int	The offset in seconds used in scheduling this block’s regularly executed task.
period	int	How often this block regularly executes its task in seconds.
memtime	int	The time, in milliseconds, used to determine if a downloaded file should be loaded into the current files cache.

Radar

These values are specific to the reflectivity Radar collector.

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
url	string	The url used to download files.
dir	string	The base directory for saving files.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
src	string	String used to create the SimpleDateFormat that is used to generate source file names.

RadarPrecip

These values are specific to the precipitation Radar collector.

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
url	string	The url used to download files.
dir	string	The base directory for saving files.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
src	string	String used to create the SimpleDateFormat that is used to generate source file names.

imrcp.collect.NDFDSky

Key Name	Type	Purpose
file	string	Source file name.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
url	string	The url used to download files.
dir	string	The base directory for saving files.
offset	int	The offset in seconds used in scheduling this block’s regularly

		executed task.
period	int	How often this block regularly executes its task in seconds.
time	int	The amount of time in seconds to go in the past to initially download files.

imrcp.collect.NDFDTd

Key Name	Type	Purpose
File	string	Source file name.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
url	string	The url used to download files.
dir	string	The base directory for saving files.
offset	int	The offset in seconds used in scheduling this block's regularly executed task.
period	int	How often this block regularly executes its task in seconds.
time	int	The amount of time in seconds to go in the past to initially download files.

imrcp.collect.NDFDTemp

Key Name	Type	Purpose
file	string	Source file name.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
url	string	The url used to download files.
dir	string	The base directory for saving files.
offset	int	The offset in seconds used in scheduling this block's regularly executed task.
period	int	How often this block regularly executes its task in seconds.
time	int	The amount of time in seconds to go in the past to initially download files.

imrcp.collect.NDFDWspd

Key Name	Type	Purpose
file	string	Source file name.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
url	string	The url used to download files.
dir	string	The base directory for saving files.
offset	int	The offset in seconds used in scheduling this block's regularly executed task.
period	int	How often this block regularly executes its task in seconds.
time	int	The amount of time in seconds to go in the past to initially download files.

imrcp.collect.KCScoutDetectors

Key Name	Type	Purpose
offset	int	The offset in seconds used in scheduling this block's regularly executed task.
period	int	How often this block regularly executes its task in seconds.
dest	string	String used to create the SimpleDateFormat that is used to

		generate destination file names.
idfile	string	The file that contains the detector mappings.
freq	int	How often a file is collected, in milliseconds, for this block.
test	string	Must be “True” or “False”, if “True” the block will run in test mode.
testfile	string	The file used in test mode.
pw	String array	List of encoded passwords to cycle through for the KCScout site
user	String array	List of usernames to cycle through for the KCScout site. Must be in corresponding order compared to the pw configuration item.

imrcp.collect.KCScoutIncidents

Key Name	Type	Purpose
offset	int	The offset in seconds used in scheduling this block’s regularly executed task.
period	int	How often this block regularly executes its task in seconds.

imrcp.store.KCScoutDetectorsStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dir	string	String used to create the SimpleDateFormat that is used to generate base directories.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
subobs	string array	Array of obs type strings that this store produces.
freq	int	How often a file is collected, in milliseconds, for this block.
delay	int	The time, in milliseconds, to takes for a file to be valid after it has been collected.
range	int	The length of time, in milliseconds, that a file is valid.
limit	int	The maximum number of files that can be loaded into the current files cache in memory for this block.
keeptime	int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.
obslength	Int	The time in seconds an observation is considered value.

imrcp.store.KCScoutIncidentsStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
subobs	string array	Array of obs type strings that this store produces.
limit	int	The maximum number of files that can be loaded into the current files cache in memory for this block.
delay	int	The time, in milliseconds, to takes for a file to be valid after it has been collected.
range	int	The length of time, in milliseconds, that a file is valid.
freq	int	How often a file is collected, in milliseconds, for this block.

lrulim	int	The maximum number of files that can be loaded into the lru in memory for this block.
offset	Int	The offset in seconds used in scheduling this block's regularly executed task.
period	Int	How often this block regularly executes its task in seconds.

IncidentDbWrapper

Key Name	Type	Purpose
tol	int	Tolerance for snapping incidents to a segment.
extend	int	The time, in milliseconds, to extend an incident if it is not closed by its estimated end time.

imrcp.store.BayesStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
subobs	string array	Array of obs type strings that this store produces.
freq	int	How often a file is collected, in milliseconds, for this block.
delay	int	The time, in milliseconds, to takes for a file to be valid after it has been collected.
range	int	The length of time, in milliseconds, that a file is valid.
limit	int	The maximum number of files that can be loaded into the current files cache in memory for this block.
keeptime	int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
lrulim	int	The maximum number of files that can be loaded into the lru in memory for this block.

imrcp.geosrv.SegmentShps

Key Name	Type	Purpose
dir	String	Directory that contains segment files.
segment	String	The file extension of any segment file the system should load in the configured directory.

imrcp.geosrv.NED

Key Name	Type	Purpose
dir	string	Directory that contains the NED files.

imrcp.imports.DataImports

Key Name	Type	Purpose
nodedbf	string	File name of the node .dbf.
nodeshp	string	File name of the node .shp.
linkdbf	string	File name of the link .dbf.
linkshp	string	File name of the link .shp.
idfile	string	File that contains the detector mappings.
length	string	Multipler used to get the length into meters.

segment	string	The segment file generated.
bridge	string	File that contains the bridge information.
tol	int	Tolerance used to snap a bridge endpoint to a segment.

imrcp.forecast.bayes.ArchiveFile

Key Name	Type	Purpose
mapping	string	File that contains detector mappings.
event	string	Directory that contains the archive event files.
detector	string	Directory that contains the archive detector data.
output	String	Name of the output file.
upbox	String	Csv list of the upper bounding box in the study area. Format is left,bottom,right,top written in integer micro degrees scaled to 7 decimal places.
lowbox	String	Csv list of the lower bounding box in the study area. Format is left,bottom,right,top written in integer micro degrees scaled to 7 decimal places.
tol	Int	Tolerance used in segment snap functions.
minmax	String	File name that contains the min and max values.
outdir	String	Directory where the output file is written to.

imrcp.forecast.bayes.RealTimeBayes

Key Name	Type	Purpose
subscribe	string array	Array of instance names this block subscribes to.
attach	string array	Array of instance names this block is dependent on.
mapping	String	File that contains the detector mappings.
highway	String	File that contains the highway segments.
period	Int	How often this block regularly executes its task in seconds.
offset	Int	The offset in seconds used in scheduling this block's regularly executed task.
forecast	Int	The number of hours to forecast.
output	String	String used to create the SimpleDateFormat that is used to generate destination file names.
freq	Int	How often a file is collected, in milliseconds, for this block.
java	String	Path to the java executable used to run the .jar.
jar	string	Path to the Bayes executable .jar.
net	String	Path to the Bayes network file.
inpath	String	The directory where the input file is located.
outpath	String	The directory where the output file is saved.
inname	String	Name of the input file.
outname	String	Name of the output file.
fcststart	Int	The hour relative to now to start creating forecasts.

imrcp.forecast.treps.DetectorMapping

Key Name	Type	Purpose
idfile	string	File that contains the detector mapping.
output	string	Path of the output file.

imrcp.forecast.treps.RealTimeDetector

Key Name	Type	Purpose

attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
output	string	Path of the output file.
period	int	How often this block regularly executes its task in seconds.
offset	int	The offset in seconds used in scheduling this block's regularly executed task.
mapping	int	File that contains the detector mapping.
rtoffset	int	The real-time offset in milliseconds.
minfile	int	The number of minutes to put in a file.

imrcp.forecast.treps.RealTimeIncident

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
output	string	Path of the output file.
period	int	How often this block regularly executes its task in seconds.
offset	int	The offset in seconds used in scheduling this block's regularly executed task.

imrcp.forecast.treps.RealTimeWorkzone

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
output	string	Path of the output file.
period	Int	How often this block regularly executes its task in seconds.
offset	Int	The offset in seconds used in scheduling this block's regularly executed task.
rate	Int	The capacity reduction rate.

imrcp.forecast.treps.RealTimeWeather

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
output	String	Path of the output file.
period	Int	The period of each forecast interval.
interval	Int	The number of forecast intervals to include in a file.

imrcp.store.ObsView

Key Name	Type	Purpose
multi	string array	Array of weather obs types that come from multiple stores.

SourceUnits

Key Name	Type	Purpose
file	string	Path of the file that contains the source unit information.

imrcp.forecast.mdss.Metro

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
fcsthrs	Int	The number of hours used to fill in the forecast arrays.

obshrs	Int	The number of hours used to fill in the observation arrays.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
threads	Int	The number of threads the metro thread pool uses.
dir	string	Directory where temporary metro files are saved.
offset	Int	The offset in seconds used in scheduling this block's regularly executed task.
period	Int	How often this block regularly executes its task in seconds.
test	string	Must be "True" or "False", if "True" the block will run in test mode.
details	string	String used to create the SimpleDateFormat that is used to generate metro detail file names.

DoMetroWrapper

Key Name	Type	Purpose
roadmin	Int	Minimum road temperature allowed in C.
roadmax	Int	Maximum road temperature allowed in C.
ssmin	Int	Minimum sub surface temperature allowed in C.
ssmax	Int	Maximum sub surface temperature allowed in C.
airmin	Int	Minimum air temperature allowed in C.
airmax	Int	Maximum air temperature allowed in C.
windmax	Int	Maximum wind speed allowed in m/s.
prmin	Int	Minimum pressure allowed in Pa.
prdef	Int	Default pressure in Pa.
prmax	Int	Maximum pressure allowed in Pa.
initrc	string array	Initial road conditions used if a past metro file is not available.
initrt	string array	Initial road temperatures used if a past metro file is not available.
dir	string	Directory where temporary metro files are saved.
precip	int	Time, in milliseconds, to use when getting precipitation amounts from the radar precip store.
raincut	String	Threshhold of the rain reservoir in mm to determine pavement state.
snowcut	String	Threshhold of the snow/ice reservoir in mm to determine pavement state.

imrcp.subs.Subscription

Key Name	Type	Purpose
subsRoot	String	Base directory where subscription information is saved.
offset	Int	The offset in seconds used in scheduling this block's regularly executed task.
period	Int	How often this block regularly executes its task in seconds.

imrcp.alert.Alerts

Key Name	Type	Purpose
freq	Int	How often a file is collected, in milliseconds, for this block.
highways	string	File that contains the highway segment ids.

WeatherAlerts

Key Name	Type	Purpose
subscribe	string array	Array of instance names this block subscribes to.

attach	string array	Array of instance names this block is dependent on.
rules	string array	Array of rule names. When an alert block starts this tells its configuration object what rule names to read.
<list of rules>	string array	Each rule's key value must be a string from the "rules" array. A rule consists of the alert type, which is a string defined in the ObsType class, and a list of conditions. A condition consists of four strings: obs type id, min value, max value, and units. Min and max values can a double, an empty string if there is not a min or max value, or a string that is a lookup string for the obs type in ObsType.
fcst	int	The number of forecast hours to create alerts.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
obs	string array	Array of obs type ids that are used by this block to create alerts.
fcstinc	int	The time, in milliseconds, of the shortest forecast interval for the observations needed to generate alerts.

TrafficAlerts

Key Name	Type	Purpose
Subscribe	string array	Array of instance names this block subscribes to.
Attach	string array	Array of instance names this block is dependent on.
Rules	string array	Array of rule names. When an alert block starts this tells its configuration object what rule names to read.
<list of rules>	string array	Each rule's key value must be a string from the "rules" array. A rule consists of the alert type, which is a string defined in the ObsType class, and a list of conditions. A condition consists of four strings: obs type id, min value, max value, and units. Min and max values can a double, an empty string if there is not a min or max value, or a string that is a lookup string for the obs type in ObsType.
Fcst	int	The number of forecast hours to create alerts.
Dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
Obs	string array	Array of obs type ids that are used by this block to create alerts.
Fcstinc	int	The time, in milliseconds, of the shortest forecast interval for the observations needed to generate alerts.

MetroAlerts

Key Name	Type	Purpose
Subscribe	string array	Array of instance names this block subscribes to.
Attach	string array	Array of instance names this block is dependent on.
Rules	string array	Array of rule names. When an alert block starts this tells its configuration object what rule names to read.
<list of rules>	string array	Each rule's key value must be a string from the "rules" array. A rule consists of the alert type, which is a string defined in the ObsType class, and a list of conditions. A condition consists of four strings: obs type id, min value, max value, and units. Min and max values can a double, an empty string if there is not a min or max value, or a string that is a lookup string for the obs type in ObsType.
Fcst	int	The number of forecast hours to create alerts.

Dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
Obs	string array	Array of obs type ids that are used by this block to create alerts.
Fcstinc	int	The time, in milliseconds, of the shortest forecast interval for the observations needed to generate alerts.

AHPSAlerts

Key Name	Type	Purpose
subscribe	string array	Array of instance names this block subscribes to.
attach	string array	Array of instance names this block is dependent on.
rules	string array	Array of rule names. When an alert block starts this tells its configuration object what rule names to read.
<list of rules>	string array	Each rule's key value must be a string from the "rules" array. A rule consists of the alert type, which is a string defined in the ObsType class, and a list of conditions. A condition consists of four strings: obs type id, min value, max value, and units. Min and max values can a double, an empty string if there is not a min or max value, or a string that is a lookup string for the obs type in ObsType.
fcst	Int	The number of forecast hours to create alerts.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
obs	string array	Array of obs type ids that are used by this block to create alerts.
fcstinc	int	The time, in milliseconds, of the shortest forecast interval for the observations needed to generate alerts.

imrcp.alert.SpeedStatsAlerts

Key Name	Type	Purpose
subscribe	string array	Array of instance names this block subscribes to.
attach	string array	Array of instance names this block is dependent on.
rule	String	A double between 0 and 1 representing the percent of excepted speed.
array	Int	The length of the arrays used to represent an alert.
file	String	Path to the file that stores all of the speed stats data.
mapping	String	Path to the file that contains detector mappings.

imrcp.alert.AlertsStore

Key Name	Type	Purpose
subobs	string array	Array of obs type strings that this store produces.
freq	Int	How often a file is collected, in milliseconds, for this block.
delay	Int	The time, in milliseconds, to takes for a file to be valid after it has been collected.
range	Int	The length of time, in milliseconds, that a file is valid.
limit	Int	The maximum number of files that can be loaded into the current files cache in memory for this block.
keeptime	Int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.
lrulim	Int	The maximum number of files that can be loaded into the lru in

		memory for this block.

WeatherAlertsStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dest	String	String used to create the SimpleDateFormat that is used to generate destination file names.
recover	String	Path to the file used to save alerts that are in memory if the system is shutdown.

TrafficAlertsStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
recover	string	Path to the file used to save alerts that are in memory if the system is shutdown.

MetroAlertsStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dest	String	String used to create the SimpleDateFormat that is used to generate destination file names.
recover	String	Path to the file used to save alerts that are in memory if the system is shutdown.

SpeedStatsAlertsStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
recover	string	Path to the file used to save alerts that are in memory if the system is shutdown.

AHPSAlertsStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dest	String	String used to create the SimpleDateFormat that is used to generate destination file names.
recover	String	Path to the file used to save alerts that are in memory if the system is shutdown.

imrcp.system.Units

Key Name	Type	Purpose
file	String	Path to the file that contains unit conversions used by the system.

imrcp.forecast.treps.OutputManager

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
offset	Int	The offset in seconds used in scheduling this block's regularly executed task.
period	Int	How often this block regularly executes its task in seconds.
dir	String	Directory where temporary treps files are saved.
network	String	Name of the network file. It is located in the "dir" directory.
destdir	String	String used to create the SimpleDateFormat that is used to generate destination file names.
date	String	String used to create the SimpleDateFormat to parse the time and date from the treps files.
time	String	The file to parse for the run time of treps.
mapping	string array	Array that contains a list of pairs. The first value of a pair is the string obs type. The second value of a pair is the treps file that contains observations of the paired obs type.

imrcp.store.TrepsStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
subobs	string array	Array of obs type strings that this store produces.
freq	int	How often a file is collected, in milliseconds, for this block.
delay	int	The time, in milliseconds, to takes for a file to be valid after it has been collected.
range	int	The length of time, in milliseconds, that a file is valid.
limit	int	The maximum number of files that can be loaded into the current files cache in memory for this block.
keeptime	int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
lrulim	int	The maximum number of files that can be loaded into the lru in memory for this block.

TrepsFtp

Key Name	Type	Purpose
ftp	string	The ip address of the ftp site.
user	string	The user name for the ftp site.
pw	string	The password for the ftp site.
dir	string	Directory where temporary Treps files are saved.
offset	int	The offset in seconds used in scheduling this block's regularly executed task.
period	int	How often this block regularly executes its task in seconds.
deftmout	Int	Default timeout for the ftp client in milliseconds.
contmout	Int	Connection timeout for the ftp client in milliseconds.

dattmout	Int	Data timeout for the ftp client in milliseconds.
files	string array	Array of files to download from the ftp site.

imrcp.forecast.treps.TrepsCollect

Key Name	Type	Purpose
offset	Int	The offset in seconds used in scheduling this block's regularly executed task.
period	Int	How often this block regularly executes its task in seconds.
test	String	Must be "True" or "False", if "True" the block will run in test mode.

DefaultTrepsCollect

Key Name	Type	Purpose
files	string array	Array of files to check if they are ready to be collected from TrepsFtp.
toffset	Int	The offset in seconds used in scheduling this block's regularly executed task if in test mode.
tperiod	Int	How often this block regularly executes its task in seconds if in test mode.

VehTrepsCollect

Key Name	Type	Purpose
files	string array	Array of files to check if they are ready to be collected from TrepsFtp.
toffset	Int	The offset in seconds used in scheduling this block's regularly executed task if in test mode.
tperiod	Int	How often this block regularly executes its task in seconds if in test mode.

imrcp.collect.AHPS

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
offset	Int	The offset in seconds used in scheduling this block's regularly executed task.
period	Int	How often this block regularly executes its task in seconds.
download	String	The url of the file to download.
url	String	The url that is polled to determine when a new file is ready to download.
tgz	String	String used to create the SimpleDateFormat that is used to generate file names for the .tgz files.
file	String	The files that contains the AHPS station and flood stage data.
freq	Int	How often a file is collected, in milliseconds, for this block.
dest	String	String used to create the SimpleDateFormat that is used to generate destination file names.
test	String	Must be "True" or "False", if "True" the block will run in test mode.
testmod	String	The timestamp used to load a file in testmod. The format is MM/dd/yyy HH:mm UTC.
field	String	Name of the field in the dbf that contains the desired data.
search	String	Name of the file to search for to check for updated timestamp.

imrcp.store.AHPSStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
freq	int	How often a file is collected, in milliseconds, for this block.
delay	int	The time, in milliseconds, to takes for a file to be valid after it has been collected.
range	int	The length of time, in milliseconds, that a file is valid.
limit	Int	The maximum number of files that can be loaded into the current files cache in memory for this block.
keeptime	Int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.

imrcp.collect.SubSurfaceTemp

Key Name	Type	Purpose
file	String	Path to the file where observations are written.
id	Int	The site id.
user	String	User name for the ftp site.
pw	String	Password for the ftp site.
offset	Int	The offset in seconds used in scheduling this block's regularly executed task.
period	Int	How often this block regularly executes its task in seconds.
ftp	String	Url of the ftp site.
init	String	The sub surface value to initially use if a value is not in the file.

imrcp.collect.CAP

Key Name	Type	Purpose
offset	Int	The offset in seconds used in scheduling this block's regularly executed task.
period	Int	How often this block regularly executes its task in seconds.
states	string array	Array of the state abbreviations that fills in part of the url to download the CAP feed.
url	String	The base url used to download files.
urlend	String	The ending of the url.
output	String	The file used to write the xml version of the CAP feeds.

imrcp.store.CAPStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
fips	string	The file that contains the fips codes and other polygon definitions used by CAP.
freq	int	How often a file is collected, in milliseconds, for this block.
delay	int	The time, in milliseconds, to takes for a file to be valid after it has been collected.

range	int	The length of time, in milliseconds, that a file is valid.
keeptime	int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.
lrulim	int	The maximum number of files that can be loaded into the lru in memory for this block.
subobs	string array	Array of obs type strings that this store produces.
limit	int	The maximum number of files that can be loaded into the current files cache in memory for this block.

Imrcp.geosrv.Polygons

Key Name	Type	Purpose
file	string	Path to the file where Polygons definitions are contained.

imrcp.collect.StormWatch

Key Name	Type	Purpose
pattern	string	The pattern used to create part of the url to download files.
format	string	String used to create the SimpleDateFormat that parses the date in the StormWatch files.
offset	int	The offset in seconds used in scheduling this block's regularly executed task.
period	int	How often this block regularly executes its task in seconds.
output	string	String used to create the SimpleDateFormat that generates output file names.
file	string	Path to the file that contains the StormWatch station information.

imrcp.store.StormWatchStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
freq	int	How often a file is collected, in milliseconds, for this block.
delay	int	The time, in milliseconds, to takes for a file to be valid after it has been collected.
range	int	The length of time, in milliseconds, that a file is valid.
keeptime	int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.
lrulim	int	The maximum number of files that can be loaded into the lru in memory for this block.
limit	int	The maximum number of files that can be loaded into the current files cache in memory for this block.
subobs	string array	Array of obs type strings that this store produces.

imrcp.store.TrvlTimeStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.

subscribe	string array	Array of instance names this block subscribes to.
dest	String	String used to create the SimpleDateFormat that is used to generate destination file names.
Freq	Int	How often a file is collected, in milliseconds, for this block.
Delay	Int	The time, in milliseconds, to takes for a file to be valid after it has been collected.
Range	Int	The length of time, in milliseconds, that a file is valid.
keeptime	Int	The time, in milliseconds, used to limit files from being loaded into the current files cache. If the file is older than the keeptime, it cannot be loaded into the current files cache. A keeptime of 0 places no limit.
Lrulim	Int	The maximum number of files that can be loaded into the lru in memory for this block.
Limit	Int	The maximum number of files that can be loaded into the current files cache in memory for this block.
subobs	string array	Array of obs type strings that this store produces.

imrcp.route.Routes

Key Name	Type	Purpose
Attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
Dest	string	String used to create the SimpleDateFormat that is used to generate destination file names.
File	string	Path to the file that contains the route information.
source	string	Path to the file that contains the Vehicle Trajectories from treps.
fcstmin	int	The number of forecast minutes in the treps files.

route

Key Name	Type	Purpose
tol	Int	Tolerance used in snapping functions when create routes.

imrcp.comp.SpeedStats

Key Name	Type	Purpose
subscribe	string array	Array of instance names this block subscribes to.
attach	string array	Array of instance names this block is dependent on.
file	String	Path to the file where all the speed stats are written.
months	Int	Number of past months to initially generate stats for.
mapping	String	Path to the file that contains detector mappings.
period	Int	How often this block regularly executes its task in seconds.
fcst	Int	Number of forecast hours.
tol	Int	Tolerance used for latitude and longitude in event queries.

imrcp.alert.Notifications

Key Name	Type	Purpose
types	string array	Array of alert types that generate notifications.
tol	Int	Tolerance used when combining locations together than have the same notification.

TrafficNotifications

Key Name	Type	Purpose

subscribe	string array	Array of instance names this block subscribes to.
attach	string array	Array of instance names this block is dependent on.
file	String	Path to the temporary file that contains the current notifications.

WeatherNotifications

Key Name	Type	Purpose
subscribe	string array	Array of instance names this block subscribes to.
attach	string array	Array of instance names this block is dependent on.
file	String	Path to the temporary file that contains the current notifications.

SpeedStatsNotifications

Key Name	Type	Purpose
subscribe	string array	Array of instance names this block subscribes to.
attach	string array	Array of instance names this block is dependent on.
file	String	Path to the temporary file that contains the current notifications.

IncidentNotifications

Key Name	Type	Purpose
subscribe	string array	Array of instance names this block subscribes to.
attach	string array	Array of instance names this block is dependent on.
file	String	Path to the temporary file that contains the current notifications.

AHPSNotifications

Key Name	Type	Purpose
subscribe	string array	Array of instance names this block subscribes to.
attach	string array	Array of instance names this block is dependent on.
file	String	Path to the temporary file that contains the current notifications.

MetroNotifications

Key Name	Type	Purpose
subscribe	string array	Array of instance names this block subscribes to.
attach	string array	Array of instance names this block is dependent on.
file	String	Path to the temporary file that contains the current notifications.

WeatherNotificationsStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dest	String	String used to create the SimpleDateFormat that is used to generate destination file names.
recover	String	Path to the file used to save notifications that are in memory if the system is shutdown.
subobs	string array	Array of obs type strings that this store produces.

TrafficNotificationsStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dest	String	String used to create the SimpleDateFormat that is used to generate destination file names.
recover	String	Path to the file used to save notifications that are in memory if the

		system is shutdown.
subobs	string array	Array of obs type strings that this store produces.

SpeedStatsNotificationsStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dest	String	String used to create the SimpleDateFormat that is used to generate destination file names.
Recover	String	Path to the file used to save notifications that are in memory if the system is shutdown.
subobs	string array	Array of obs type strings that this store produces.

IncidentNotificationsStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dest	String	String used to create the SimpleDateFormat that is used to generate destination file names.
recover	String	Path to the file used to save notifications that are in memory if the system is shutdown.
subobs	string array	Array of obs type strings that this store produces.

AHPSNotificationsStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dest	String	String used to create the SimpleDateFormat that is used to generate destination file names.
recover	String	Path to the file used to save notifications that are in memory if the system is shutdown.
subobs	string array	Array of obs type strings that this store produces.

MetroNotificationsStore

Key Name	Type	Purpose
attach	string array	Array of instance names this block is dependent on.
subscribe	string array	Array of instance names this block subscribes to.
dest	String	String used to create the SimpleDateFormat that is used to generate destination file names.
recover	String	Path to the file used to save notifications that are in memory if the system is shutdown.
subobs	string array	Array of obs type strings that this store produces.

NotificationServlet

Key Name	Type	Purpose
test	String	Must be “True” or “False”, if “True” the block will run in test mode.

Appendix C. Dynasmart-X Real-Time Dynamic Traffic Assignment System

Environment Settings

Installation of TAO

DYNASMART-X is a distributed traffic estimation and prediction (TrEPS) software tool that consists of several different functional modules. All the components within it are brought together through CORBA, which is a protocol/specification/standard that allows different pieces within the software to communicate with each other. Within DYNASMART-X, CORBA is implemented using TAO, which is open source free software developed by Dr. Douglas Schmidt's team and written in C++.

In order to run DYNASMART-X successfully on a machine, TAO needs to be installed before installing DYNASMART-X. To learn more about CORBA and TAO, please visit the site: http://www.cs.wustl.edu/~schmidt/TAO.html.

1. Download ACE + TAO package (Version 6.0.1) from website: http://download.dre.vanderbilt.edu/previous_versions/.

   File name: **ACE+TAO-6.0.1.zip**, released date: 28-Jan-2011 07:03, file size: 63M

2. Unzip the ACE+TAO package, and place the unzipped folder 'ACE_wrappers' under path C:\
3. Set up system environment variables in Windows.

   Right click 'Computer' → 'Properties' → 'Advanced system settings' → 'Environment Variables'

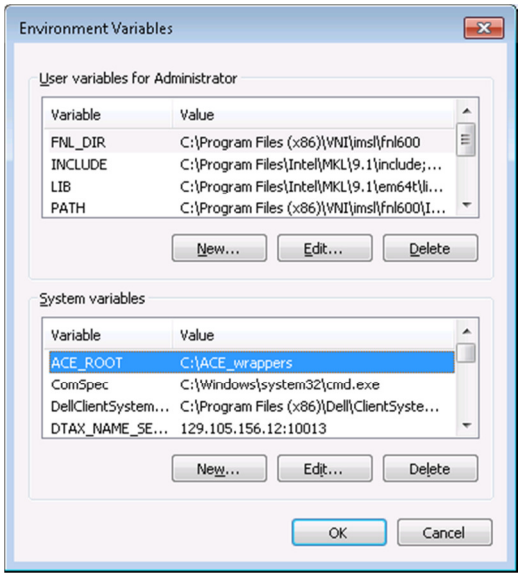

Figure 7. Windows Environment Variables window

Under 'System Variables', Click 'New' and add the following variables with their corresponding values.

Table 4. Environment Variables for TAO

Variable Name	Variable Value
ACE_ROOT	C:\ACE_wrappers
TAO_ROOT	%ACE_ROOT%\TAO
PATH (add values on the right)	%ACE_ROOT%; %ACE_ROOT%\TAO; %ACE_ROOT%\bin; %ACE_ROOT%\lib; %ACE_ROOT%\TAO\orbsvcs

Remarks: when adding variables into PATH, there should be no spaces between variable names, i.e., it should be:
%ACE_ROOT%;%ACE_ROOT%\TAO;%ACE_ROOT%\bin;%ACE_ROOT%\lib;%ACE_ROOT%\TAO\orbsvcs

4. Create a header file under %ACE_ROOT%\ace, naming as "config.h", with contents as below:

```
//------------------
// @ file config.h
#include "ace/config-win32.h"
//------------------
```

5. Open TAO_ACE_vc8.sln under %ACE_ROOT%\TAO
    a. Remarks: This solution corresponds to Visual Studio 2005; for other versions of Visual Studio (e.g. 2008 - TAO_ACE_vc9.sln, 2010 - TAO_ACE_vc10.sln), other solutions will be used.
6. Visual Studio 2005 Settings
    a. Go to 'Tools' → 'Options' → 'Projects and Solutions' → 'VC++ Directories'
        i. Choose Platform 'x64'
        ii. Add "$(PATH)" to the Include files set
        iii. Add "$(PATH)" to the Library files set
        iv. Add "$(PATH)" to the Source files set
    b. Go to 'Project' → 'Properties' → Solution Property page
        i. Choose Configuration "Active(Debug)"
        ii. Choose Platform "Active(x64)"
7. Compile. Build solution 'TAO_ACE_vc8' under both '**Debug**' mode and '**Release**' mode on 'x64' platform. After building is completed, close Visual Studio.
8. Generate ACE's MFC library
    a. Download ActivePerl for Windows 64-bit System from: http://www.activestate.com/activeperl/downloads.
    b. Install ActivePerl with Default settings.

c. Open Command window. Type “Perl -version” in the Command window to check if Perl is installed successfully. Current version will be shown as Figure 8, if Perl is installed.

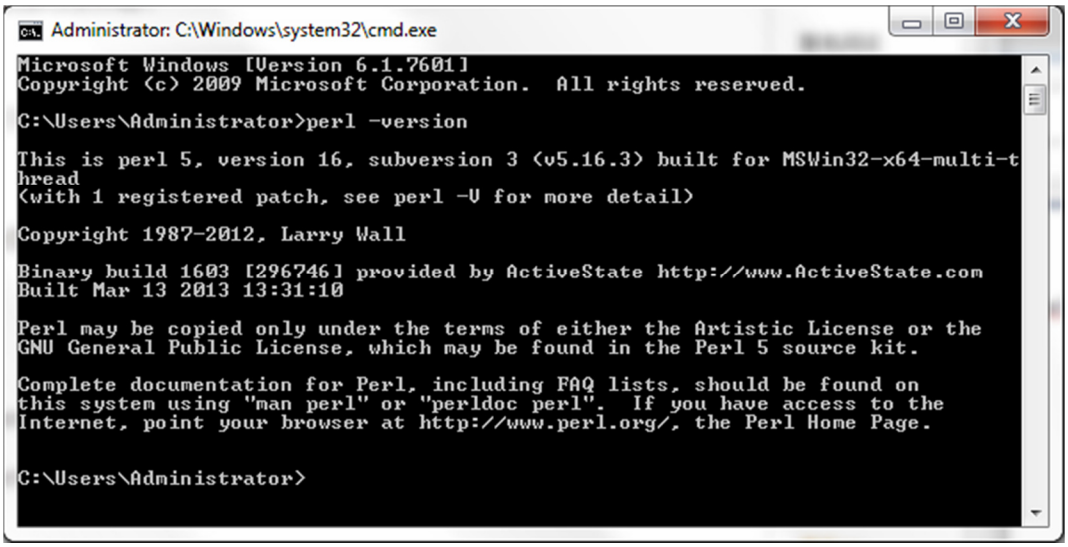

Figure 8. ActivePerl Version

d. Generate solution with MFC configuration

   i. Make sure Visual Studio is closed before generating MFC solutions.

   ii. Use command window, change path to %ACE_ROOT%\TAO (e.g., C:\ACE_wrappers\TAO).

e. Enter the following command (Figure 9): **perl c:\ace_wrappers\bin\mwc.pl –type vc8 –value_template “configurations = ‘MFC Release’ ‘MFC Debug’ Release Debug” –name_modifier “*_mfc_vc8” tao_ace.mwc**

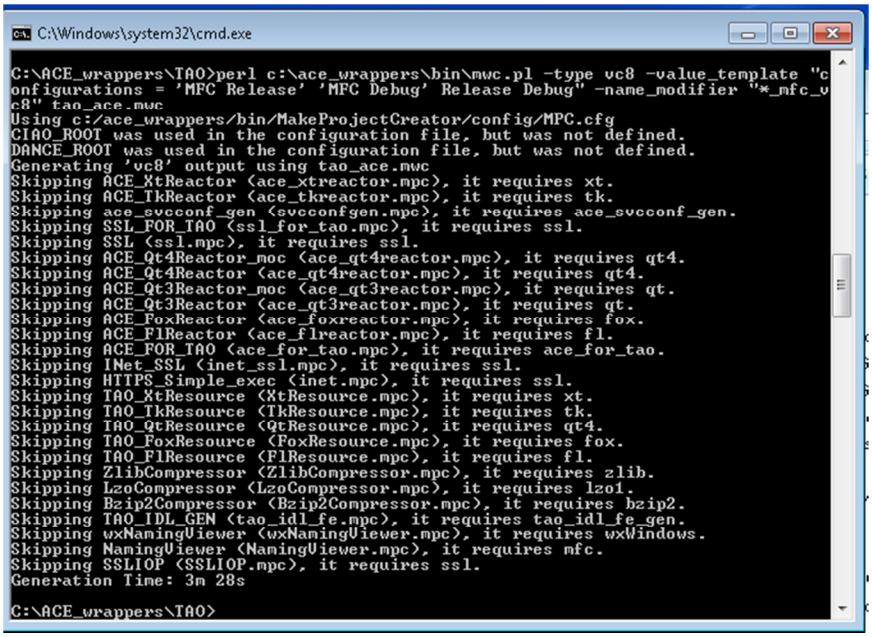

Figure 9. MFC Generation Process

f. A new solution (tao_ace_mfc_vc8.sln) will be generated with MFC configurations (Figure 10).

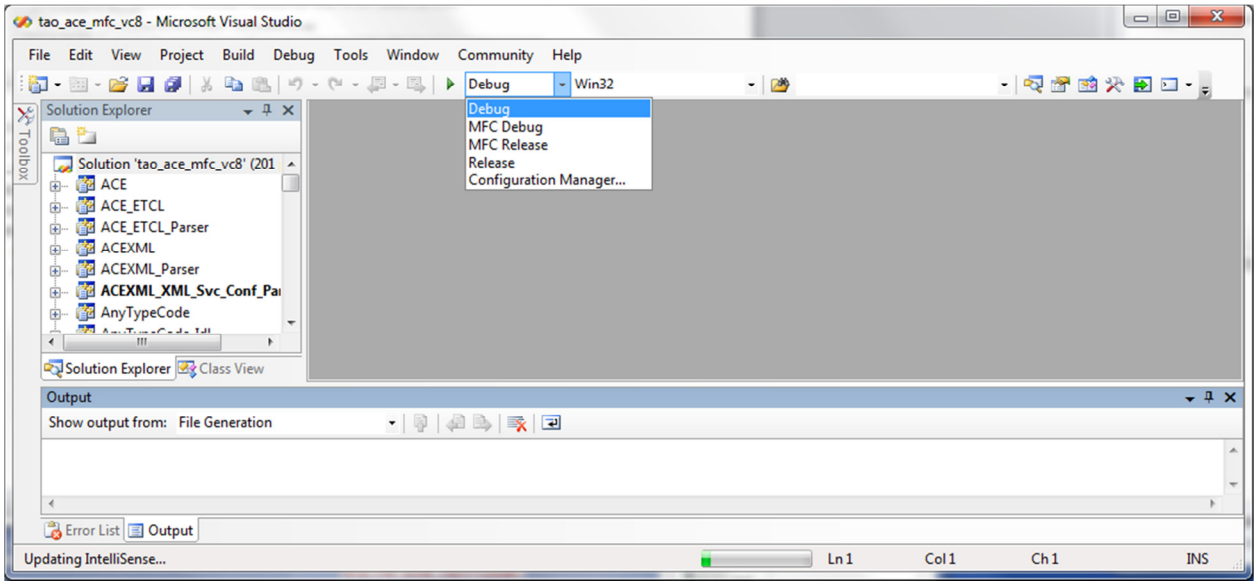

Figure 10. MFC solution window

9. Compile ACE_TAO solution under MFC.
   a. Open **'tao_ace_mfc_vc8.sln'**, unload project 'TAO_IDL_EXE'. After unloading, there will be **200** projects in total.
   b. Set the following Visual Studio environmental variables under path:
   c. Tools → Options → Projects and Solutions → VC++ Directories
   d. These changes are made on 'x64' platform.

Table 5. ACE_TAO Visual Studio Setting

Library Files	$(ACE_ROOT)\lib
Include Files	$(ACE_ROOT) $(TAO_ROOT) $(TAO_ROOT)\orbsvcs
Executable Files	$(ACE_ROOT)\bin

**Note that in some cases the order of the files has influence to compilations. Try to move the added file up or down if a problem occurs in installation.*

   e. Open **'TAO_ACE_vc8.sln'**, rebuild 'TAO_IDL_EXE' and 'Naming Service' projects under both 'Release' and 'Debug' configurations, on the 'x64' platform.
   f. Open **'tao_ace_mfc_vc8.sln'**, build the entire solution under both 'MFC Release' and 'MFC Debug' configurations, on the 'x64' platform.

10. After the compilation of TAO under the four configurations is finished, the ACE library folder (C:\ACE_wrappers\lib) will contain four kinds of files. Take the object File Library *TAO_AnyTypeCode*.lib* as follows:
    - TAO_AnyTypeCode.lib is generated by Release mode
    - TAO_AnyTypeCoded.lib is generated by Debug mode
    - TAO_AnyTypeCodemfc.lib is generated by MFC Release mode
    - TAO_AnyTypeCodemfcd.lib is generated by MFC Debug mode

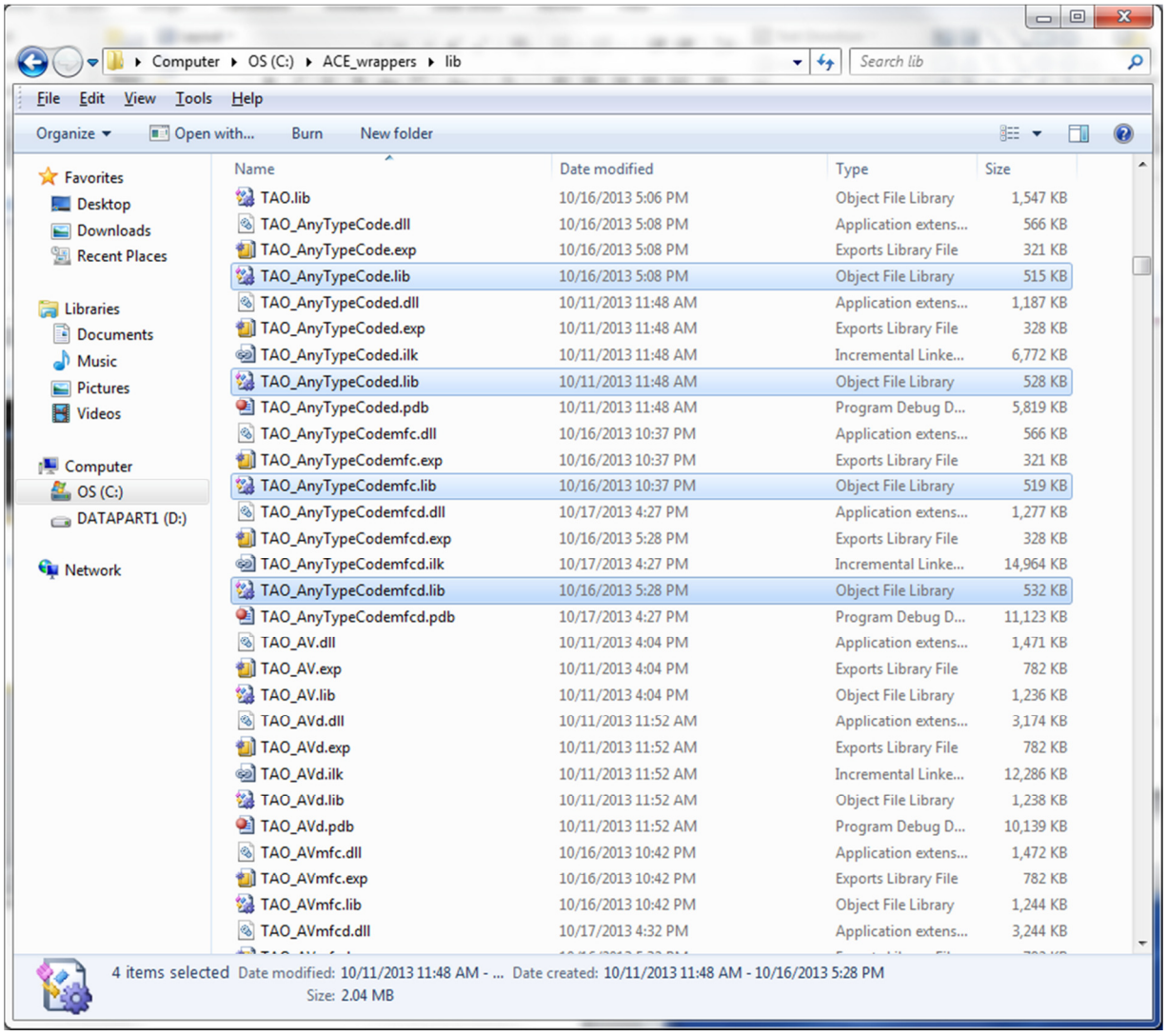

Figure 11. Library Files Generated

Installation of Python

1. Download Python 2.7 or Python 3.6.
2. Install the packages required: datetime, numpy, shutil and urllib.

Environment Variable Setting for DYNASMART-X

Set the following two Windows Environment Variables (naming service host and DTAX run directory):

- DTAX_NAME_SERVICE_HOST
  – IP address of your machine :10013 (e.g. 129.105.156.12:10013)
- DTAX_RUN_DIR
  – c:\DTAX\RUN

Tutorial of Dynasmart-X

The purpose of this tutorial section is to assist users in getting started with the DYNASMART-X system. In this tutorial, a procedure is described to illustrate how to run all the modules of DYNASMART-X.

Input Data

DYNASMART-X requires several input files for all the modules. These input files reside in the /DTAX/run, /DTAX/run/estimate, /DTAX/run/predict, /DTAX/run/odestimate, /DTAX/run/odpredict, /DTAX/run/stcc, and /DTAX/run/ltcc. The provided folders already include all the required input files.

Details of the DYNASMART input are described in two technical memoranda developed for the IMRCP project and provided separately:

1. Northwestern University Transportation Center; *Integrated Modeling for Road Condition Prediction – DYNASMART TrEPS Input Files Description*; Technical Memorandum; June 28, 2016.
2. Northwestern University Transportation Center; *Integrated Modeling for Road Condition Prediction – DYNASMART Input Files Descriptio – II*; Technical Memorandum; December 15, 2016.

Please note that in this project, /DTAX/run/predict1 will not be used, but the files placed in /DTAX/run/predict need to be copied to /DTAX/run/predict1 to initiate the model.

Initializing DYNASMART-X

Go to C:\DTAX\run, find the python script – TrEPS.py. Double click this file to load input files and initiate dsxgui.exe.

The window will open as shown below. It is comprised of a pull-down menu, toolbar, simulation view, clock view, and information view.

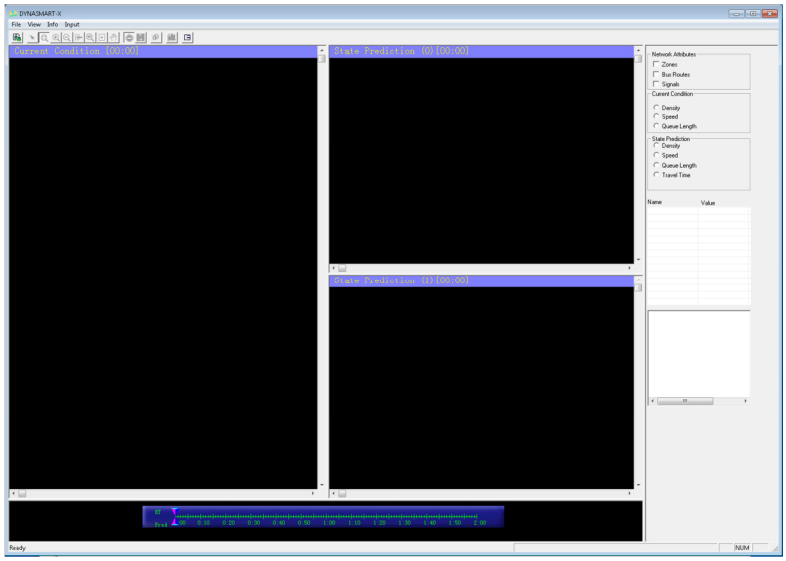

Figure 12. Initial GUI of DYNASMART-X

To initiate DTAX, go to “File” on the DYNASMART-X window menu, then to “Load Data”, and click “1 Check System Settings”. When it shows the window shown in Figure 13, choose “Real Time Mode” instead of “Sequential Mode”.

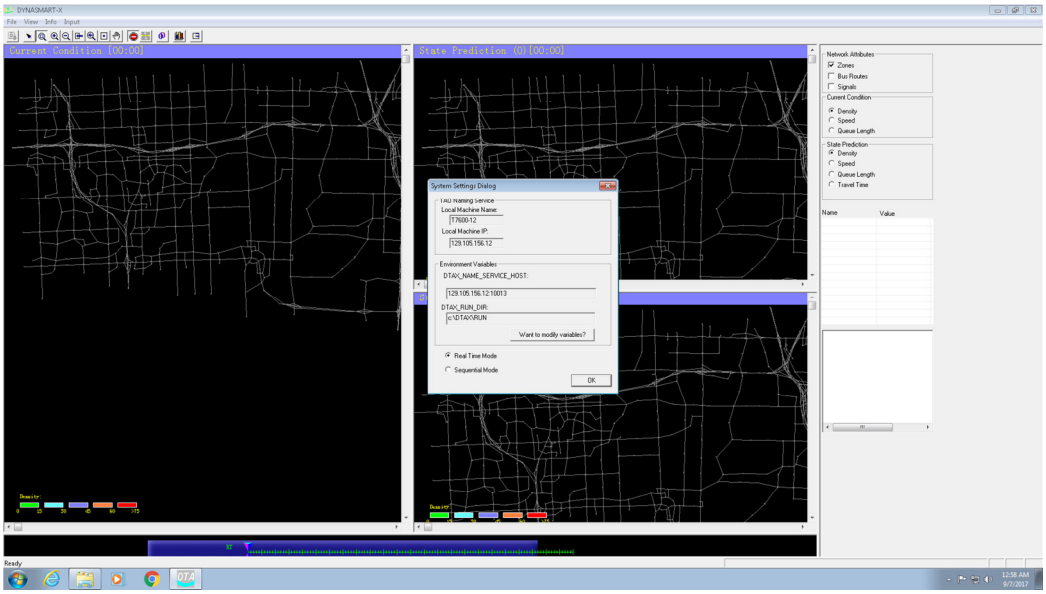

Figure 13. DYNAMSMART Mode Selection Window

Next, click “2 Naming Service”. The window shown in Figure 14 will open.

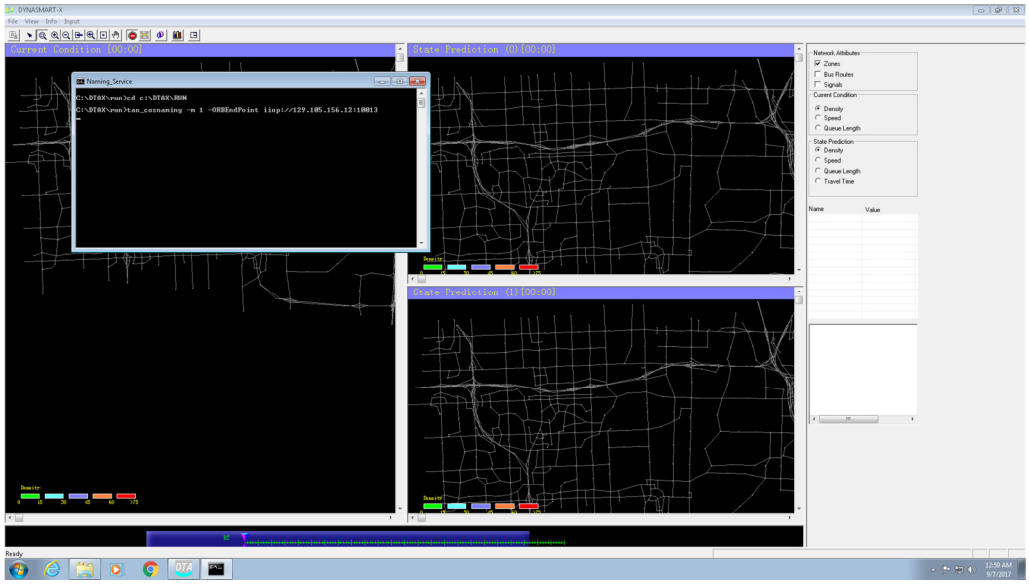

Figure 14. DYNASMART Naming Service Window

Skip the third item in the menu list and click “4 Active Models on Server Machine”. It will initiate several modules as Figure 15 shows.

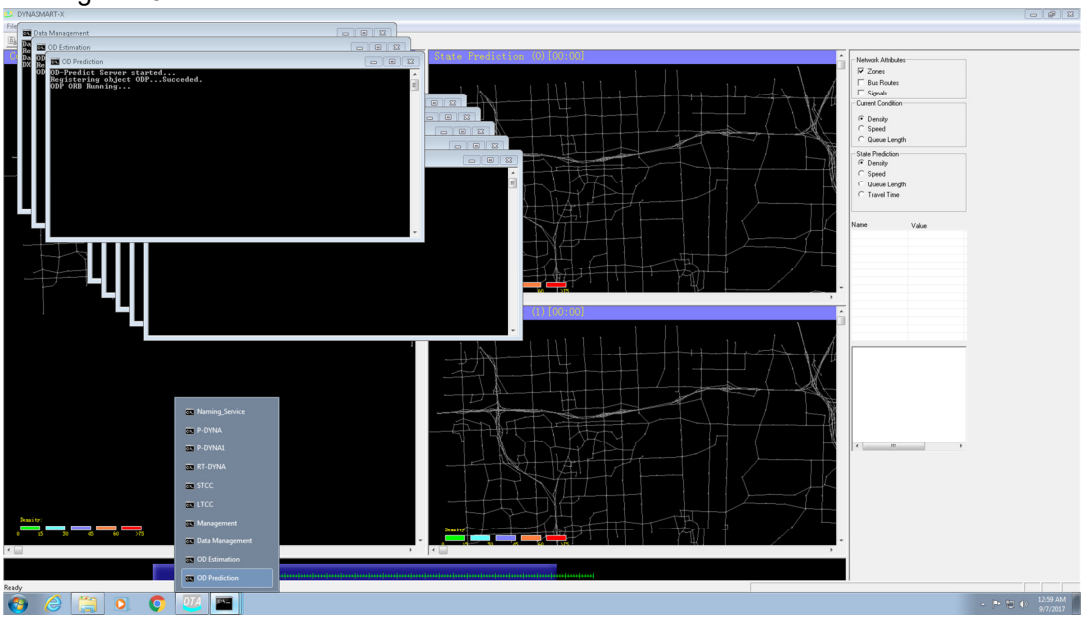

Figure 15. DYNASMART Open Services Windows

Then, set up CORBA and start DYNASMART-X. CORBA requires the correct installation of TAO as described earlier. If the model is initiated correctly, the window will appear, as shown in Figure 16. The right bottom box shows what module is currently running in the model.

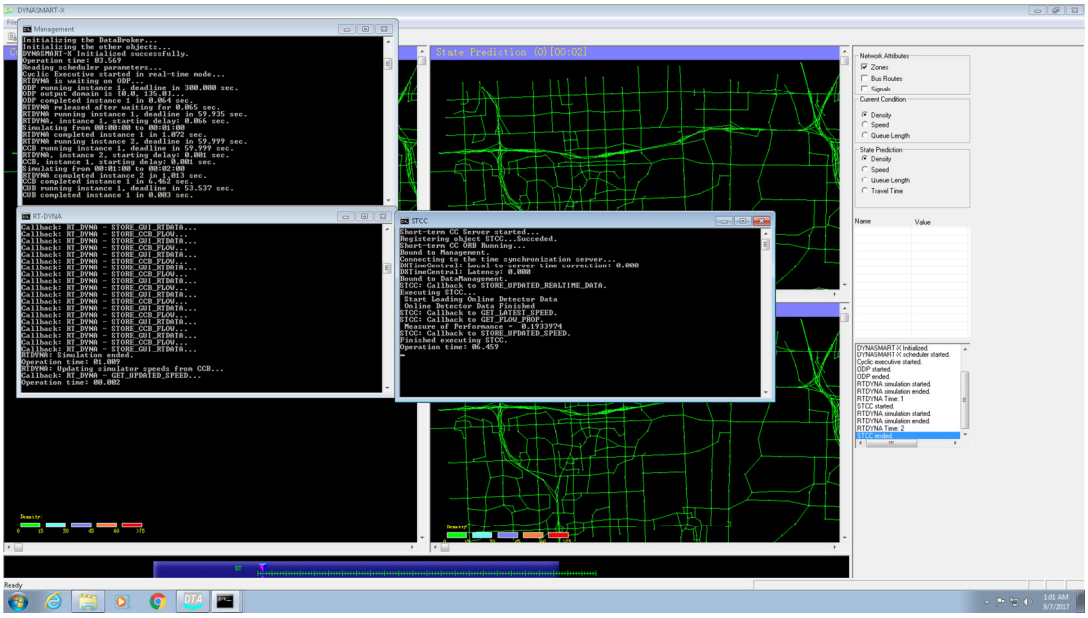

Figure 16. Completed DYNASMART Initialization

Appendix D. Glossary

AHPS	**Advanced Hydrologic Prediction Service**
ConOps	**Concept of Operations**
CSV	**comma separated values**
CV	**Connected Vehicles**
FHWA	**Federal Highway Administration**
IMRCP	**Integrated Modeling for Road Condition Prediction**
ITS	**Intelligent Transporation Systems**
JNI	**Java Native Interface**
MDSS	**Maintenance Decision Support System**
METRo	**Model of the Environment and Temperature of Roads**
NUTC	**Northwestern University Transportation Center**
NWS	**National Weather Service**
OSM	**OpenStreetMap**
RWF	**Road Weather Forecast**
RWIS	**Road Weather Information System**
RWMP	**Road Weather Management Program**
SHP	**shapefile**
SSD	**solid state disc**
TrEPS	**Traffic Estimation and Prediction System**
TSMO	**Transportation system management and operations**
USDOT	**United States Department of Transportation**